Probably the most convenient feature of the OKA (both versions) is the ability to use the previous solution as the first guess for the next solution. This situation occurs typically in water resources networks; the same network has to be solved for many time intervals and usually the only parameters that change are the upper and lower bounds of some arcs. These changes are normally within 50% of their previous value. In this situation the OKA for one interval is started from flows and marginal costs obtained in the the previous interval and this makes the execution much faster, since the first guess about flows and marginal costs is already very close to an optimal solution.

4.3 Flowchart of Barr's Algorithm

The steps of Barr's algorithm are summarized in the flowchart on the following pages. Before the beginning of execution according to this flowchart, the following must be done:

1) The original capacitated network must be transformed into a corresponding pseudo network. This means that each pseudo arc must have its number, net capacity, marginal cost and head and tail nodes.
2) Label eligibility of all pseudo arcs must be examined and sets P_i and Q_i initialized for all nodes in the network.

Having completed this the execution can start. Four distinct locations are marked with A, B, C and D in the flowchart and they represent the following:

A - beginning of the algorithm

B - flow change section

C - labelling procedure

D - potential change procedure

Although all major phases of Barr's algorithm have already been mentioned in the previous pages of this chapter, the following are pointed out:

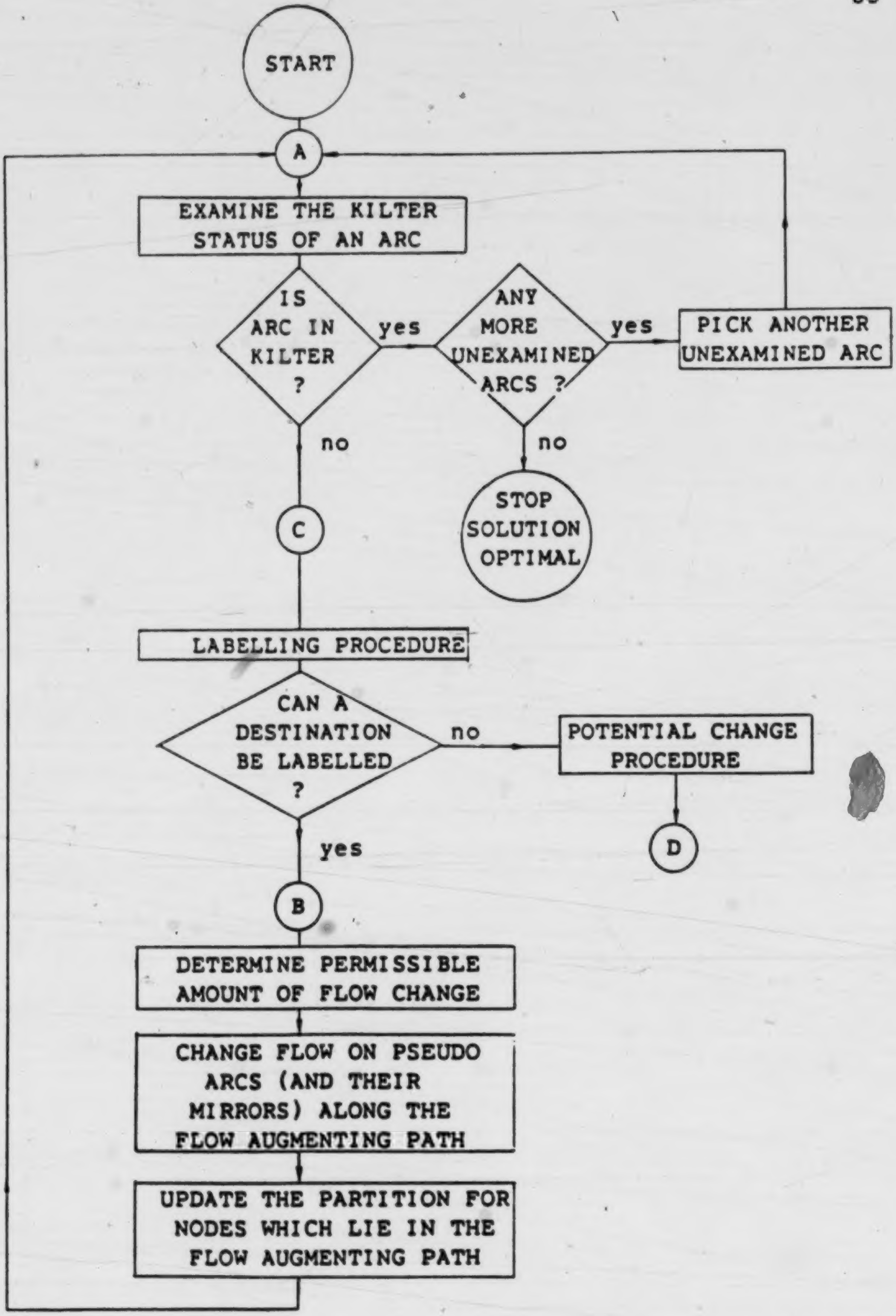
a) The order in which arcs are examined in the beginning of the program has no significance. The only important thing is to avoid examining the same arcs more than once.

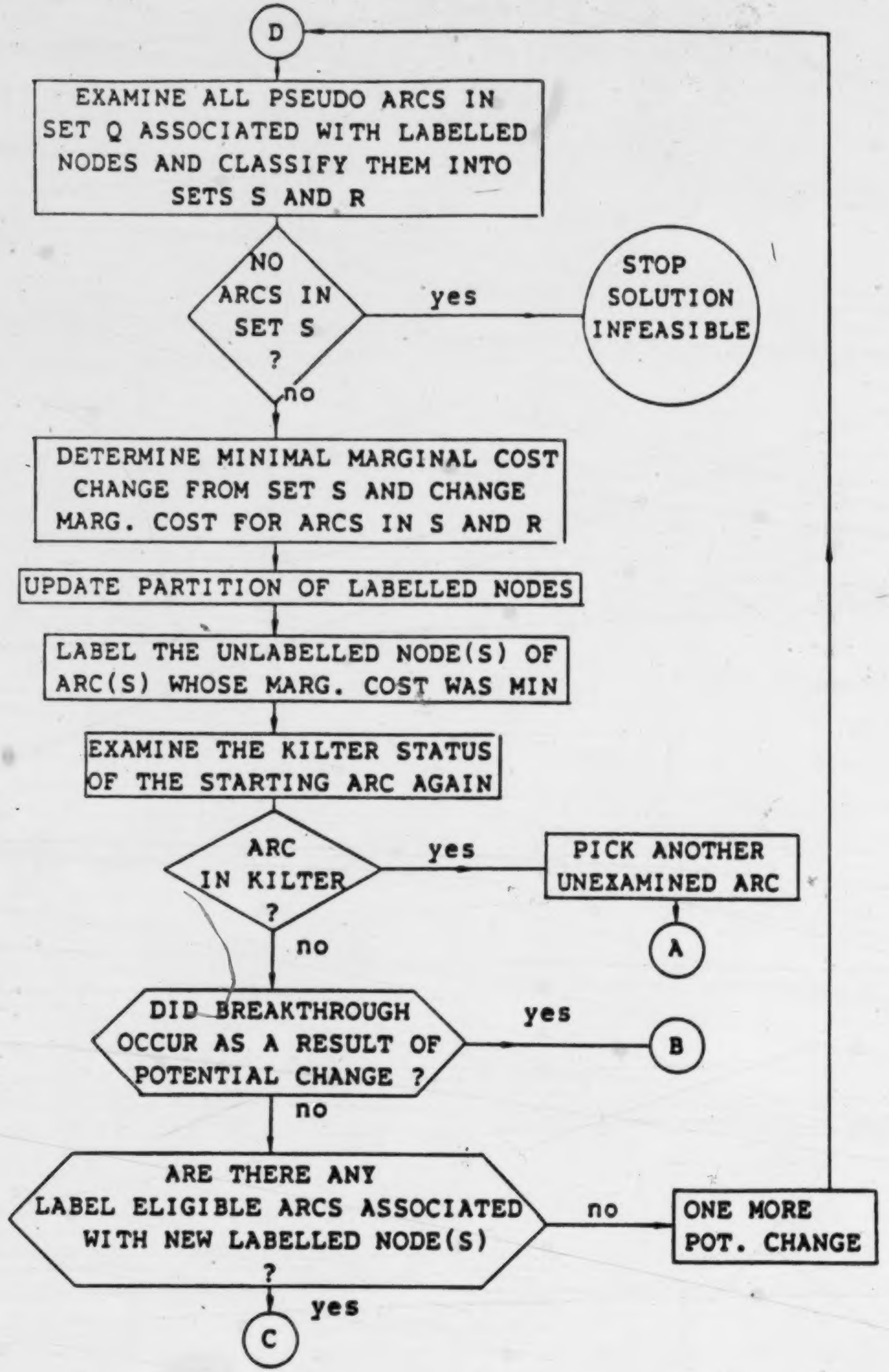
b) Potential change section contains two steps that might need further comments:

- "Examine the kilter status of the st~~art~~ing arc again". In order to appreciate this step, one has to realize that the starting arc might belong to set Q_i of the starting node (origin). In this case the marginal cost of the starting arc will be changed during potential change and this change might bring this arc in-kilter. This causes the algorithm to return to section A on the flowchart and continue search for other out-of-kilter arcs.

- "Did breakthrough occur as a result of potential change?"

To demonstrate this possibility, go back to fig 3.1 and imagine that labelling procedure in corresponding pseudo network ended at nodes 8, 9, 6 and 5. The next step leads to the potential change procedure. Examine all arcs in set Q_i for all labelled nodes (in this case nodes 0, 3, 8, 9, 7, 6 and 5). It might happen that pseudo arc in set S with its marginal cost closest to zero is arc (5,n). After potential change this arc becomes label eligible thus completing the path with the starting arc. In this case there is no need to go back to the labelling procedure as a flow augmenting path is already available (fig 3.2); therefore, simply proceed with flow change (section B).


START
A
EXAMINE THE KILTER STATUS OF AN ARC
IS ARC IN KILTER ?
yes
ANY MORE UNEXAMINED ARCS ?
yes
PICK ANOTHER UNEXAMINED ARC
no
no
STOP SOLUTION OPTIMAL
C
LABELLING PROCEDURE
CAN A DESTINATION BE LABELLED ?
no
POTENTIAL CHANGE PROCEDURE
D
yes
B
DETERMINE PERMISSIBLE AMOUNT OF FLOW CHANGE
CHANGE FLOW ON PSEUDO ARCS (AND THEIR MIRRORS) ALONG THE FLOW AUGMENTING PATH
UPDATE THE PARTITION FOR NODES WHICH LIE IN THE FLOW AUGMENTING PATH


D
EXAMINE ALL PSEUDO ARCS IN SET Q ASSOCIATED WITH LABELLED NODES AND CLASSIFY THEM INTO SETS S AND R
NO ARCS IN SET S ?
yes
STOP SOLUTION INFEASIBLE
no
DETERMINE MINIMAL MARGINAL COST CHANGE FROM SET S AND CHANGE MARG. COST FOR ARCS IN S AND R
UPDATE PARTITION OF LABELLED NODES
LABEL THE UNLABELLED NODE(S) OF ARC(S) WHOSE MARG. COST WAS MIN
EXAMINE THE KILTER STATUS OF THE STARTING ARC AGAIN
ARC IN KILTER ?
yes
PICK ANOTHER UNEXAMINED ARC
A
no
DID BREAKTHROUGH OCCUR AS A RESULT OF POTENTIAL CHANGE ?
yes
B
no
ARE THERE ANY LABEL ELIGIBLE ARCS ASSOCIATED WITH NEW LABELLED NODE(S) ?
no
ONE MORE POT. CHANGE
yes
C

5. DETAILED DESCRIPTION OF THE NEW OPTIMIZATION SUBPROGRAM IN THE WRMM

5.1 General Features

Two different optimization subprograms based on Barr's algorithm were developed and tested for implementation in the WRMM. They differ in the way they utilize the existing data from the WRMM. Two parameters were considered for the final choice:

- execution times
- memory increase requirement

Since both versions showed approximately the same execution times, the version that was finally adopted was the one which requires less total memory increase. This is a significant factor because of the latest tendency to install and run WRMM on microcomputers.

Variables of the final version are listed alphabetically in Appendix I, while Appendix II contains source code listing of the main program (WRMM) and optimization subroutines for both versions. Names of the subroutines and their functions are identical in both versions and they are given in the following table:

SUBROUTINE	*FUNCTION*
PSARCS	Reformulates the original capacitated network into a pseudo network.
ARRAYS	Initial setup of sets P and Q for all nodes in the network.
KILTER	Derives an optimal solution or declares the problem infeasible. Corresponds to the flowchart on pages 42-43 in the previous section.
LBLRTN	Attempts to label a flow augmenting path ("label routine").
PINQ	Moves a pseudo arc from subset P into subset Q.
QINP	Moves a pseudo arc from subset Q into subset P.

Subroutines PSARCS, ARRAYS and KILTER are called by the main program; PSARCS is called only once, while ARRAYS and KILTER are called up to five times within one time interval. Subroutines LBLRTN, PINQ and QINP are called by the KILTER subroutine. The total number of calls to these subroutines depends on two factors:

- Size of the network (number of arcs).
- Starting flows and marginal cost of all arcs.

The latter is also referred to as the "first guess"; flows do not have to be between the bounds but conservation of flow has to be satisfied for every node. The closer the first guess is to an optimal solution, the less work is needed to bring all arcs in-kilter, which decreases the number of calls to subroutines LBLRTN, PINQ and QINP.

Subroutine KILTER has two returns to the main program, depending on the type of solution obtained (optimal or infeasible). In case of infeasibility, in order to isolate the infeasible arc the flows of all arcs previously brought in-kilter. have to be retrieved. This is done before returning to the main program and the abnormal termination section in the main program then calls subroutine ARCCHK which isolates the out-of-kilter arc and gives possible reasons for infeasibility.

Subroutine LBLRTN also has two optional returns to KILTER subroutine, depending on whether labelling procedure resulted in a breakthrough or non-breakthrough.

All arguments passed between the subroutines and calling programs are passed in an argument list. There are two reasons for this:

1) Adjustable dimensions make it easier to install WRMM on a micro computer, since changes are needed in the main program only.
2) WRMM is compiled using the IL(DIM) FORTRAN VS compiler option. This option speeds up the processing of the data passed in an argument list, which are now read in-line, rather than by a library call. To verify this, two versions of the same program (with common statements and without) were tested on large files. It was found that

the version with the IL(DIM) option is just as fast as the version with the common statement.

All phases of the new OKA are demonstrated in the following numerical example.

5.2 Numerical Example

Consider the network in fig 5.1 with six nodes and ten arcs. Note the following:

1) Each arc is oriented.
2) The network is circulatory.
3) Each arc has its upper bound, lower bound and cost.

Also the upper bound is greater than or equal to the lower bound for each arc in the network.

The following table gives the tail node, head node, upper bound, lower bound and cost for each arc of the network in fig 5.1:

arc number	tail node	head node	upper bound	lower bound	arc cost
1	1	2	6	0	9
2	3	1	3	1	2
3	1	4	5	0	0
4	2	3	4	2	1
5	4	3	7	0	0
6	5	2	5	1	2
7	3	5	8	1	5
8	5	4	6	0	4
9	6	5	7	0	0
10	4	6	5	3	3

This network will be solved from an initial assumption that all flows are zero and all marginal costs are equal to the actual costs (i.e. all node potentials equal to zero in the original OKA).

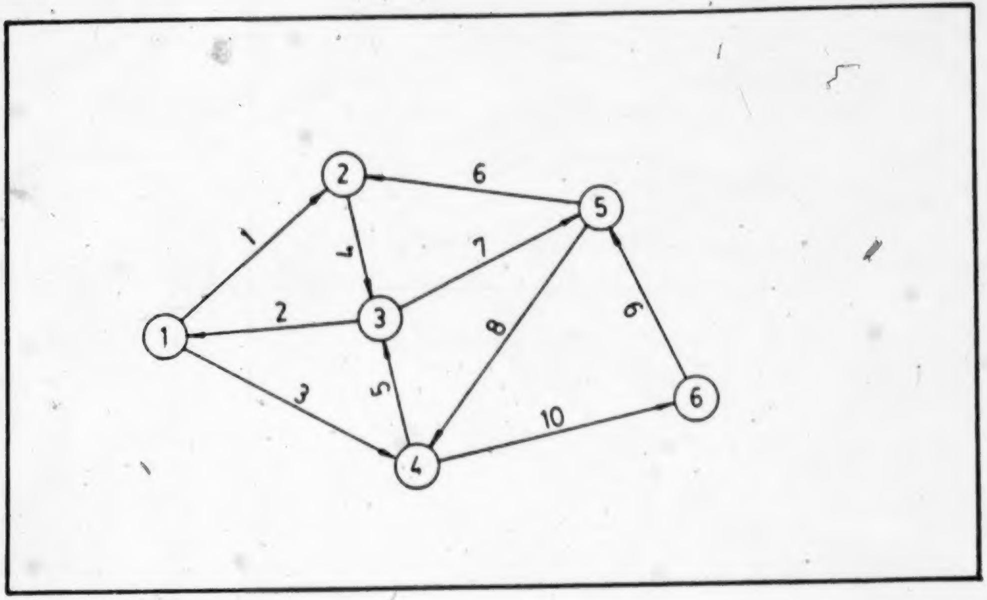


fig 5.1 Simple circulatory network

First create a pseudo network as shown in the fig 5.2. All main pseudo arcs retain the same direction and arc number; each mirror pseudo arc has the opposite direction and number equal to the number of the main pseudo arc plus ten.

Table 5.1 shows the head node, net capacity (PANC) and marginal cost (PACOST) for all pseudo arcs in the network. Partition of pseudo arcs into sets P_i and Q_i is given by arrays NDARCS ("node-arcs"), NODE and MIDL.

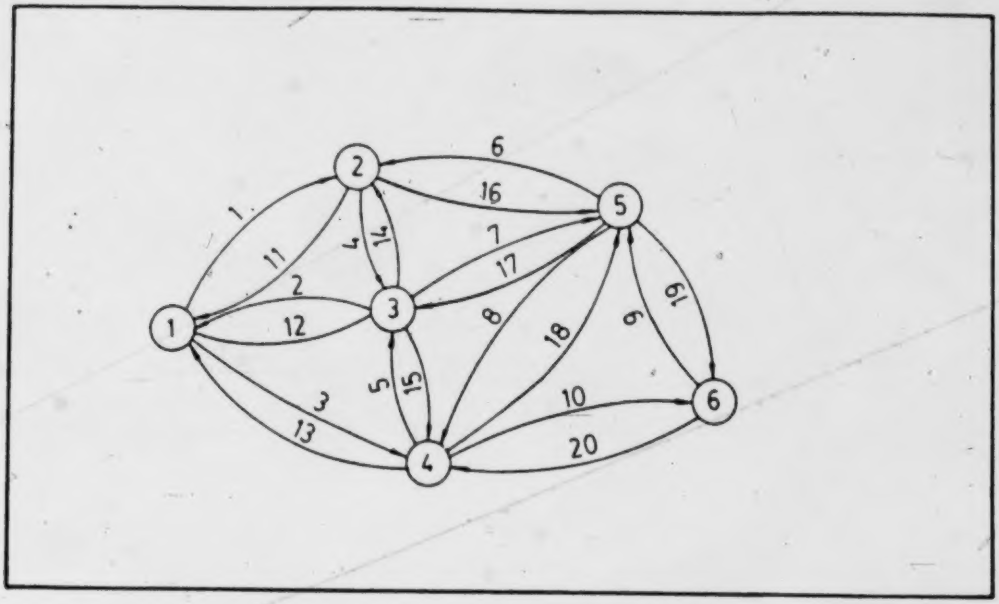


fig 5.2 Pseudo network

Array NDARCS is obtained by considering all outgoing pseudo arcs for node i and placing their arc numbers in array NDARCS in such a way that all label eligible pseudo arcs come first. Array NODE splits NDARCS array into subsets of outgoing arcs for each node. For example, NODE(3.3) indicates the first position for a pseudo arc directed out of node 3, while NODE(4)-1 indicates the last. For example, node 3 in fig 5.2 has four outgoing pseudo arcs (arbitrary order): 2, 7, 15 and 14. This information can be retrieved by recalling the values from NDARCS(NODE(3.3)) to NDARCS(NODE(4)-1). Note that NODE(4) gives the position of the first outgoing pseudo arc for node 4. Thus the previous position in array NDARCS is the last which belongs to node 3. Subpartition of array NDARCS into label eligible and ineligible arcs is given by the value in array MIDL, which points to the position of the last label eligible arc of a subset in NDARCS. For example, for node 3 the appropriate value of array MIDL is 8, which means that pseudo arcs in position 7 and 8 are label eligible. In other words, arc numbers stored in NDARCS(3.7) and NDARCS(3.8.1) indicate the label eligible outgoing pseudo arcs for node 3, while the rest of them are label ineligible (NDARCS(3.8.2) and NDARCS(3.9)).

To summarize, arcs located between NDARCS(NODE(I)) and NDARCS(MIDL(I)) constitute set P_i, while arcs located between NDARCS(MIDL(I)+1) and NDARCS(NODE(I+1)-1) constitute set Q_i. Partition is thus extended for each node i in the network. Once array NODE is determined it remains constant since the physical network structure does not change. However, arrays NDARCS and MIDL will undergo changes as pseudo arcs change their label eligibility. Changes of NDARCS and MIDL will be demonstrated in the following example.

Having completed reformulation of the original network and initial setup of the partition for all the nodes, the execution can start following the flowchart in the previous chapter.

Starting from pseudo arc 1, examine its kilter status. This arc is found to be in-kilter, since its mirror capacity and marginal cost satisfy condition (3.23'). Examine the kilter status of the mirror arc 11. This arc is also in-kilter (condition (3.25')). [*]

Pick another arc and examine its kilter status. It can be any arc in the network, therefore following the way in which arcs are numbered constitutes no loss of generality.

Arc 2 does not satisfy any of the conditions (3.23')-(3.25') and it is therefore out-of-kilter; the desired action is an increase of flow (equivalent to decrease of net capacity). Start the labelling procedure in an effort to isolate a flow augmenting path. Since the head node of pseudo arc 2 is node 1, examine its set P_1. There is one label eligible arc in P_1 and that is arc 3. Therefore, in the first cycle node 4 is labelled with number 3, which should be symbolically understood as: "it is possible to get to 4 along route 3". Labelling procedure now proceeds from node 4 and two pseudo arcs (5 and 10) are found in set P_4. As soon as node 3 is labelled, the labelling procedure stops since the destination is labelled. The flow augmenting path consists of arcs 2, 3 and 5. Arcs 3 and 5 are in-kilter and arc 2 will determine the amount of net capacity change (equal to 1) for all other arcs on the flow augmenting path. Net capacities of pseudo arcs 2, 3, and 5 are reduced by one and 12, 13 and 15 are increased by one. This automatically brings arcs 2 and 12 in kilter. Also, net capacity of arc 13 has become positive, so this arc becomes label eligible, and it is shifted from Q_4 into P_4 as shown in Table 5.2.

---

[*] This is equivalent to arc 1 in the original network being in-kilter, which is true, since its flow equals the lower bound and marginal cost are positive. Therefore, each time an arc in the original network is in-kilter, both corresponding pseudo arcs are also in-kilter.

The execution continues by examining the kilter status of other arcs. Arcs 3 and 13 are in-kilter, and the next out-of-kilter arc encountered is arc 4. Labelling procedure starts from node 3 and results in a breakthrough. This time the flow augmenting path consists of arcs 4, 7 and 6. Net capacities of these arcs (and their mirrors) are changed by 1. The new net capacity of arc 16 becomes zero and it becomes label ineligible. Consequently, it is moved from set P_5 into Q_5. Results of this are shown in table 5.3. However, this was not enough to bring arc 4 in-kilter. The algorithm now starts another labelling procedure which begins from node 3. In the first cycle node 4 was labelled. There are three label eligible arcs in set P_4. However, one of them leads to node 3, which is already labelled. The other two lead to nodes 1 and 6, and these nodes are labelled accordingly. Finally, in the third cycle node 5 is labelled starting from node 6. There are no label eligible arcs left and the destination (node 2) hasn't been reached. The labelling procedure now stops and potential change procedure is applied in order to extend the labelling tree.

The potential change procedure starts with isolating all arcs which lead from labelled to unlabeled nodes. In this case only node 2 is unlabeled and the arcs which lead to node 2 are arcs 14, 1 and 6. These arcs are now classified into sets S and R; arcs 1 and 6 belong to set S (their net capacity and marginal cost are positive), while arc 14 belongs to set R. The marginal cost of arc 6 is the closest to zero and all arcs in sets S and R (and their mirrors) will have their marginal cost changed by 2. Marginal cost of arc 6 thus becomes zero and it becomes label eligible. Therefore, arc 6 is moved back into P_5. New marginal costs and the updated partition are shown in table 5.4. Node 2 is now labelled as a result of the potential change and this time potential change procedure has resulted in a breakthrough. The flow augmenting path consists of arcs 6, 9, 10, 15 and 4. Net capacities of these arcs (and their mirrors) are changed by 1 unit (directed by arc 15) and this finally brings arc 4 in-kilter. Arc 15 becomes label ineligible and arc 19 becomes label eligible, since its new net capacity becomes positive. New net capacities and partition are shown in table 5.5.

The algorithm then continues search for other out-of-kilter arcs. All arcs are found in-kilter except arc 10. If it is possible to bring this arc in-kilter then all arcs would be in-kilter and an optimal solution is found.

Labelling procedure now starts from node 6. Node 5 is labelled in the first cycle and node 2 in the second; after this there are no more label eligible arcs and labelling procedure stops. Potential change procedure starts with isolating arcs whose tail node is labelled and head node unlabeled, and classifying them into sets S and R. This time, arcs 20, 17 and 11 are in set R and arcs 8 and 4 are in set S. Marginal cost of arc 4 is closer to zero and arcs

20, 17, 11, 8 and 4 (and their mirrors) will have their marginal cost changed by 3. Arc 4 becomes label eligible as a result of this potential change and node 3 is labelled. New marginal costs and partition are shown in table 5.6. Since set P_3 is currently empty, there is no way to extend the labelling tree starting from this node. This is the situation when more than one subsequent potential change is needed in order to bring the starting arc (3.9) in-kilter. All pseudo arcs currently in sets S and R are now recycled as follows:

a) Arc 4 is purged from set S since it became label eligible in the previous potential change.

b) Arc 17 is purged from set R since its head node was labelled in the previous potential change.

c) Finally, arcs in set Q_3 are classified into sets S and R and added to the existing arcs in S and R.

As a result of the foregoing changes, arcs in set R are now 20, 11 and 15 while set S consists of arcs 8 and 2. These arcs (and their mirrors) will have their marginal cost changed by 1 unit (directed by arc 8). New marginal costs and partition are in table 5.7. Arc 5 becomes ineligible, while arc 8 becomes label eligible, thus completing the flow augmenting path which now consists of arcs 8, 9 and 10. Net capacities of these arcs and their mirrors are changed by 2 units (directed by arc 20) and the final result of primal and dual variables and partition is in table 5.8.

All arcs are now in-kilter and solution can be converted into solution for original network using one of the equations for net capacity. This solution is displayed in Table 5.9. The original in-kilter conditions can be easily recognized for each arc.

ARCNO	NHEAD	PANC	M.COST	NDARCS
1	2	6	9	3
2	1	3	2	12
3	4	5	0	1
4	3	4	1	4
5	3	7	0	16
6	2	5	2	11
7	5	8	5	2
8	4	6	4	7
9	5	7	0	15
10	6	5	3	14
11	1	0	-9	5
12	3	-1	-2	10
13	1	0	0	18
14	2	-2	-1	13
15	4	0	0	6
16	5	-1	-2	19
17	3	-1	-5	17
18	5	0	-4	8
19	6	0	0	9
20	4	-3	-3	20

NODE(1)= 1	MIDL(1)= 1
NODE(2)= 4	MIDL(2)= 4
NODE(3)= 7	MIDL(3)= 8
NODE(4)=11	MIDL(4)=12
NODE(5)=15	MIDL(5)=15
NODE(6)=19	MIDL(6)=19

Table 5.1 Primal and dual values and P and Q sets.

ARCNO	NET CAPACITY	MARG. COST	NDARCS
1	6	9	3
2	2	2	12
3	4	0	1
4	4	1	4
5	6	0	16
6	5	2	11
7	8	5	15
8	6	4	7
9	7	0	2
10	5	3	14
11	0	-9	5
12	0	-2	10
13	1	0	13
14	-2	-1	18
15	1	0	6
16	-1	-2	19
17	-1	-5	17
18	0	-4	8
19	0	0	9
20	-3	-3	20

```
NODE(1)=  1  MIDL(1)=  1
NODE(2)=  4  MIDL(2)=  4
NODE(3)=  7  MIDL(3)=  8
NODE(4)= 11  MIDL(4)= 13
NODE(5)= 15  MIDL(5)= 15
NODE(6)= 19  MIDL(6)= 19
```

Table 5.2 Primal and dual values and P and Q sets

ARCNO	NET CAPACITY	MARG. COST	NDARCS
1	6	9	3
2	2	2	12
3	4	0	1
4	3	1	4
5	6	0	16
6	4	2	11
7	7	5	15
8	6	4	7
9	7	0	2
10	5	3	14
11	0	-9	5
12	0	-2	10
13	1	0	13
14	-1	-1	18
15	1	0	6
16	0	-2	19
17	0	-5	17
18	0	-4	8
19	0	0	9
20	-3	-3	20

NODE(1)= 1 MIDL(1)= 1
NODE(2)= 4 MIDL(2)= 4
NODE(3)= 7 MIDL(3)= 7
NODE(4)= 11 MIDL(4)= 13
NODE(5)= 15 MIDL(5)= 14
NODE(6)= 19 MIDL(6)= 19

Table 5.3 Primal and dual values and P and Q sets

```
ARCNO  NET CAPACITY MARG. COST  NDARCS

  1          6           7         3
  2          2           2        12
  3          4           0         1
  4          3           3         4
  5          6           0        16
  6          4           0        11
  7          7           5        15
  8          6           4         7
  9          7           0         2
 10          5           3        14
 11          0          -7         5
 12          0          -2        10
 13          1           0        13
 14         -1          -3        18
 15          1           0         6
 16          0           0        19
 17          0          -5        17
 18          0          -4         8
 19          0           0         9
 20         -3          -3        20

          NODE(1)=  1  MIDL(1)=  1
          NODE(2)=  4  MIDL(2)=  4
          NODE(3)=  7  MIDL(3)=  7
          NODE(4)= 11  MIDL(4)= 13
          NODE(5)= 15  MIDL(5)= 15
          NODE(6)= 19  MIDL(6)= 19
```

Table 5.4 Primal and dual values and P and Q sets

ARCNO	NET CAPACITY	MARG. COST	NDARCS
1	6	7	3
2	2	2	12
3	4	0	1
4	2	3	16
5	7	0	4
6	3	0	11
7	7	5	15
8	6	4	7
9	6	0	2
10	4	3	14
11	0	-7	5
12	0	-2	10
13	1	0	13
14	0	-3	18
15	0	0	6
16	1	0	19
17	0	-5	17
18	0	-4	8
19	1	0	9
20	-2	-3	20

NODE(1)= 1 MIDL(1)= 1
NODE(2)= 4 MIDL(2)= 4
NODE(3)= 7 MIDL(3)= 6
NODE(4)= 11 MIDL(4)= 13
NODE(5)= 15 MIDL(5)= 16
NODE(6)= 19 MIDL(6)= 19

Table 5.5 Primal and dual values and P and Q sets

ARCNO	NET CAPACITY	MARG. COST	NDARCS
1	6	10	3
2	2	2	12
3	4	0	1
4	2	0	16
5	7	0	4
6	3	0	11
7	7	8	15
8	6	1	7
9	6	0	2
10	4	6	14
11	0	-10	5
12	0	-2	10
13	1	0	13
14	0	0	18
15	0	0	6
16	1	0	19
17	0	-8	17
18	0	-1	8
19	1	0	9
20	-2	-6	20

```
NODE(1)=  1  MIDL(1)=  1
NODE(2)=  4  MIDL(2)=  5
NODE(3)=  7  MIDL(3)=  6
NODE(4)= 11  MIDL(4)= 13
NODE(5)= 15  MIDL(5)= 16
NODE(6)= 19  MIDL(6)= 19
```

Table 5.6 Primal and dual values and P and Q sets

ARCNO	NET CAPACITY	MARG. COST	NDARCS
1	6	11	3
2	2	1	12
3	4	0	1
4	2	0	16
5	7	1	4
6	3	0	11
7	7	8	15
8	6	0	7
9	6	0	2
10	4	7	14
11	0	-11	13
12	0	-1	10
13	1	0	5
14	0	0	18
15	0	-1	6
16	1	0	19
17	0	-8	8
18	0	0	17
19	1	0	9
20	-2	-7	20

```
NODE(1)=  1  MIDL(1)=  1
NODE(2)=  4  MIDL(2)=  5
NODE(3)=  7  MIDL(3)=  6
NODE(4)= 11  MIDL(4)= 12
NODE(5)= 15  MIDL(5)= 17
NODE(6)= 19  MIDL(6)= 19
```

Table 5.7 Primal and dual values and P and Q sets

```
ARCNO  NET CAPACITY MARG. COST  NDARCS

  1          6           11          3
  2          2            1         12
  3          4            0          1
  4          2            0         16
  5          7            1          4
  6          3            0         11
  7          7            8         15
  8          4            0          7
  9          4            0          2
 10          2            7         14
 11          0          -11         13
 12          0           -1         18
 13          1            0         10
 14          0            0          5
 15          0           -1          6
 16          1            0         19
 17          0           -8          8
 18          2            0         17
 19          3            0          9
 20          0           -7         20

            NODE(1)=  1  MIDL(1)=  1
            NODE(2)=  4  MIDL(2)=  5
            NODE(3)=  7  MIDL(3)=  6
            NODE(4)= 11  MIDL(4)= 12
            NODE(5)= 15  MIDL(5)= 17
            NODE(6)= 19  MIDL(6)= 19
```

Table 5.8 Primal and dual values and P and Q sets

```
FINAL FLOWS AND MARGINAL COSTS

 AFLOW( 1)= 0   PACOST( 1)= 11
 AFLOW( 2)= 1   PACOST( 2)=  1
 AFLOW( 3)= 1   PACOST( 3)=  0
 AFLOW( 4)= 2   PACOST( 4)=  0
 AFLOW( 5)= 0   PACOST( 5)=  1
 AFLOW( 6)= 2   PACOST( 6)=  0
 AFLOW( 7)= 1   PACOST( 7)=  8
 AFLOW( 8)= 2   PACOST( 8)=  0
 AFLOW( 9)= 3   PACOST( 9)=  0
 AFLOW(10)= 3   PACOST(10)=  7
```

Table 5.9 Primal and dual values and P and Q sets

5.3 Specific Features of the WRMM Internal Network

The WRMM internal network [39], [41] has some important properties which significantly affect the solution times. These properties are the following:

1) The average number of arcs associated with a node exceeds ten.
2) Arcs leading from node i to node j are classified in arc sets.
3) About half of all arcs in the network are associated with the system balance node.

The most important feature is that all arcs belong to arc sets. The number of arcs in one set can vary between one and nine; in most of the cases it ranges from three to seven.

This gave rise to constructing such OKA subroutines which are able to utilize this information in labelling and potential change procedure. For example, consider five arcs leading from node i to node j. In the labelling procedure, node j is labelled along the first arc (i,j). If other four arcs are label eligible the algorithm will attempt to label node j four more times, and each time this attempt will result in failure since node j is already labelled. However, if all five arcs are placed in an arc set, once the head node of such arc set is labelled all other checks (for each arc within the set) can be omitted. A similar improvement is available in the potential change procedure in order to determine arcs in sets S and R. In this case, if both node i and j are labelled, there is no need to check the head node of each arc in the particular set and determine what is already known (i.e. the head node of each arc in the set is labelled).

To implement this data organization scheme in the new OKA, it has to be successfully incorporated with the existing partitioning technique.

One way is to associate the partition of pseudo arcs with arc sets, rather than with nodes, since the WRMM internal network already contains the information about arc sets.

Two versions of the KILTER subroutines differ in the following:

a) Version 1, in which the partiton is associated with pseudo arcs directed out of a node.

b) Version 2,in which all pseudo arcs directed out of a node are placed in arc sets, and these sets are partitioned into subsets of label eligible and ineligible arcs.

Along with positive effects on execution times, some negative effects were also encountered; both are briefly listed:

Positive effects

1) The new organization scheme speeds up the potential change procedure. This is also true for labelling procedure when there are one or more label eligible arcs in each arc set.

2) Subroutines PINQ and QINP execute faster, especially for arcs associated with the system balance node. This is because the appropriate arc (to be shifted from P to Q or from Q to P) is identified by knowing which arc set it belongs to. However, subroutines PINQ and QINP are usually called only once during the labelling or potential change procedures, which reduces the effect of this improvement.

Negative Effects

1) The amount of data passed between the subroutines and their calling programs is considerably increased. In addition to the existing argument list, arrays NLOWST, NHIST, NEXNOD, SETMAP, CHMAP and NSETA are passed to subroutine KILTER. Also, arrays MIDL and NODE are

increased in dimension equal to the number of arc sets (642), rather than the number of nodes (102).

2) After conducting a number of experimental runs, it was found that the number of label eligible arcs is approximately 10% of all arcs in the network. This was very detrimental for efficiency of the source code. Consider the following example: Fifteen arcs are associated with a node and they belong to five arc sets. Only one arc is label eligible. In the labelling procedure of the Barr's algorithm this arc would be quickly accessed in the appropriate set P_i. However, set P_i is now split into five subsets. Although in four of them there are no label eligible arcs, that fact is yet to be determined during execution. Thus five checks are needed instead of one, which is a serious departure from achieving higher efficiency.

In conclusion, the execution times were compared for both versions and they were found to be almost identical. The latest version showed somewhat faster execution for medium size networks, while the source code based on Barr's algorithm was slightly faster on largest networks. However, the differences do not exceed 2% in either case, and the version finally adopted is the one based entirely on Barr's idea of partitioning, since it requires less memory increase.

6. CHANGES IN THE NEW VERSION OF WRMM

6.1 Changes in Memory Requirements

The only negative effect following the implementation of the new OKA subroutines in the WRMM is that they require an increase in memory. This happens as a result of the following factors:

a) Creating pseudo network demands doubling of the previous dimension of arrays NHEAD and NTAIL, in order to make it possible to determine the head node and tail node of each pseudo arc. Strictly speaking, only one of these two arrays has to be determined for all pseudo arcs, since the other one can always be found using the mirror function. However, this was avoided since the frequent use of the mirror function would somewhat slow down the execution.

b) New primal and dual variables are associated with each pseudo arc. The node potentials array is not needed any more, but this does not constitute great savings, since the number of nodes in the network is only one tenth of the number of arcs.

c) Finally, the partitioning technique requires an additional number of arrays which were not needed in the original OKA subroutines.

All new arrays which are introduced to the WRMM as a result of implementing the new OKA subroutines, or whose dimension had to be increased thereafter are listed below:[5]

INLOC(102)
JNODE(2000)
LABEL(102)
MIDL(102)
NHEAD(2000)
NTAIL(2000)
NODE(103)
NDARCS(2000)
PANC(2000)
PACOST(2000)
VNODE(102)

Conversely, a number of arrays were deleted from the WRMM. These arrays are: ARCKST(1000), LARC(102), LNODE(102), LFLOW(102) and PI(102).

Thus the actual increase in memory might seem significant, but when considered as a percentage of the total it represents only a 3.7% increase.

6.2 Changes in the Main Program and Other Subroutines

As a result of implementation of the new OKA, the WRMM underwent the following changes:

a) *Initialization Section in the Main Program*

- Arrays NHEAD and NTAIL are initialized with their new dimension
- Array LABEL and variable INFES are initialized with zero
- Constants R and Z are assigned their values

Apart from this, variables MND1 and MARCS2 have to be initialized, but this is possible only after the internal network is created. These two variables are thus initialized prior to calling the subroutine PSARCS, which is placed immediately before the main loop.

---

[5] Only arrays NHEAD and NTAIL existed in the old version of the WRMM with a dimension of 1000.

b) *Changes in the Subroutine ARCCHK*

Subroutine ARCCHK prints the upper bound, flow, lower bound and arc cost for all arcs in the network. This subroutine is not called unless it was specified by the user in the input data file. However, in the case of an infeasible solution this subroutine is called automatically and its function is to isolate the out-of-kilter arc and to help determine the possible causes of infeasibility.

Changes in this subroutine are related to the removal of the array ARCKST, by which the kilter status of an arc was easily inspected (in the old version). Variable ARCKST could have only three values: zero, minus one and one. In the case of zero, the appropriate arc was in-kilter. In case of minus one or one, the arc was out-of-kilter and the desired action was the decrease or increase of flow, respectively.

The search for an out-of-kilter arc in the new version is conducted using the logical "if" statements which check for the compliance with conditions (3.23)-(3.25). For this purpose array PACOST is passed to the subroutine from the main program.

c) *Other Changes*

The values of the upper bounds of some arcs were set to 10^{32} in the old version. This caused problems in the conversion of flows to net capacities and back, since some numbers were too big and the system would automatically ignore some digits which were out of range. In order to avoid this problem, all upper bounds which were previously set to 10^{32} were reset to 10^{4}. Since upper bounds are in CMS units, it is estimated that this is just as good a representation for the unlimited upper bounds of any natural system.

A considerable effort was involved in an attempt to install new OKA subroutines with integer primal and dual variables. There are two reasons for this:

1) Integer numbers are processed faster
2) Integer numbers require less memory
3) Adding or subtracting integers has no truncation error as is the case with floating point numbers

This idea was successfully utilized in case of the PACOST array. However, changing net capacity into INTEGER*4 variable caused many problems and it was finally decided to leave it as REAL*8. The problems were mainly related to the lack of accuracy due to the conversion. Due to the truncation error associated with floating point numbers, a zero limit of 10^{-7} was introduced. Also most of the "if" statements in subroutines PASSTR, CMSMTR and MTRCMS had to be "relaxed" by a zero tolerance of 10^{-7}. In other words, any value that falls between -10^{-7} and 10^{-7} is considered equal to zero.

6.3 Comparison of Results

Once finally debugged, the new version of the WRMM gave slightly different results than the old version for some simulation runs. The best way to compare two different solutions is to compare the value of their objective functions:

$$\Sigma c^{+}_{i,j} \cdot x_{i,j} , \qquad \text{for all arcs } (i,j)$$

The solution which minimizes the objective function can be considered more optimal. Thus a "do" loop which calculates the objective function was inserted right after the kilter subroutines in both versions. In the case of small and medium size networks the objective functions yielded almost identical values. The differences occurred in some time intervals, but they were usually far below 0.1%. However, in the case of the largest network (South Saskatchewan ten year simulation run) the objective functions differed up to 3% in some time intervals. However, it was still impossible to determine which solution is more optimal, since these differences oscillate from time interval to time interval, to the advantage of one version or the other.

As part of the investigation of this phenomena it was further found that the new OKA subroutines yield different solutions if pseudo arcs in sets P_i are shuffled. Note that the sequence of placing label eligible arcs into sets P_i was never given any importance. The effect of this can be observed in fig 3.1. One recalls that in the original example, starting from node t node 3 was labelled first and then node 7. This was due to the position of arcs 14 and 18 in set P_t. If their positions are reversed, node 7 is labelled first and then node 3 in the first cycle. In the second cycle P_7 is examined first and node 9 is labelled by arc 12, which is a now a different label than previously. This may eventually result in a different flow augmenting path, different amount of flow change and different setup of sets S and R in the potential change. Therefore, slight differences in results for some time intervals are due to millions of combinations on the route to an optimal solution.

Practically, these differences are of no particular importance. It was found that the differences in the objective function are due to the different flows in arcs whose cost per unit of flow is very low (mainly one or two). Converting these low penalties to zero would likely establish more stable optimality. Another action that would also stabilize the optimality is assigning different penalties in the input data file to all arcs in the network. However, this is neither justifiable nor practical, especially for large networks.

6.4 Comparison of Execution Times

Three different files were used to time the execution of the new WRMM. In order to assess the actual effectiveness of the new KILTER subroutines a new system timer was used, with the accuracy of one millionth of a second. This system timer was used to measure the portion of cpu time required by the kilter subroutines only.

All runs were executed on the IBM 370 Mainframe in the Government of Alberta Terrace Computing Center. The BOWOP85 simulation run was used as a typical small to medium size network. It consists of 25 nodes and 176 arcs, and the simulation run was set for one year with forty one time intervals. The same input data file was used to run ten and twenty five year simulation runs.

Table 6.1 CPU TIME FOR BOWOP85 SIMULATION RUNS

	1 year		10 years		25 years	
	old	new	old	new	old	new
TOTAL CPU	3.37	2.92	33.28	28.10	86.76	69.84
KILTER CPU	1.17	0.72	14.45	9.27	38.16	21.24

The foregoing table reveals that the improvements are not that great for a network of this size. This is basically due to the small portion of the total cpu time used by the kilter subroutines (between 30% and 40%) and more importantly, the reduced effect of the improvements within the new OKA itself on a network of this size. In fact, Barr's paper emphasizes the number of nodes as a dominant factor in computational improvements. Since there can be a maximum of 102 nodes in the internal network of WRMM, it was not expected that any improvements would exceed a 40% total reduction.

The other two files that were tested were large. Files RUN1A1 and RUN1A10 represent the South Saskatchewan simulation run for one year and ten years. For the RUN1A10 run, one year data is not simply repeated ten times but each year has different data. The internal network consists of 95 nodes and 792 arcs. File MOR7 is somewhat more complex. It has 28 years of actual data comprising the driest and wettest years, and the number of arcs is slightly increased due to the introduction of control structures for all reservoirs. Due to the greater variation of the input data from time interval to time interval, and more reiterations within one time interval (caused by the increased number of control structures), MOR7 simulation run is much harder to execute. The execution times are shown in the following table:

Table 6.2 CPU TIMES FOR SOUTH SASKATCHEWAN SIMULATION RUNS

	RUN1A1		RUN1A10		MOR7	
	old	new	old	new	old	new
TOTAL CPU	8.70	3.82	87.50	30.03	413.63	224.25
KILTER CPU	6.43	1.51	73.47	16.01	250.78	61.40

For large networks, the portion of cpu required by the subroutines is between 60% and 85%, and the kilter subroutines are executed four to five times faster. As a result of this, total cpu time is reduced two to three times. This is a significant achievement since large simulation runs account for 90% of the computational cost related to the use of the WRMM.

6.5 FURTHER ENHANCEMENTS OF BARR'S ALGORITHM

Apart from the more efficient data structure discussed earlier, other improvements could be implemented and same ideas in this respect are briefly outlined in the following.

One of them is to try to implement a global strategy within the algorithm. It can be noted that, during labelling procedure, many closed paths of label eligible arcs are isolated, but the current algorithm ignores them unless the starting out-of-kilter arc is included. Therefore, if all the network is scanned and all possible cycles isolated, a finite number of flow changes could take place, followed by a finite number of potential changes, which would generate several label eligible arcs at once, instead of only one. This approach would be extremely complicated in the case of existing array data organization. However, with more efficient data organization (discussed in the following section) this global approach could be applied much easier and the effects of better algorithm and better data structure would be cumulative.

There is a lot of inertia associated with using FORTRAN, partly because many professionals are familiar with it and have no time to achieve similar excellence in other programming languages, and partly because hu[illegible] programs have been written in FORTRAN and their rewriting to another language might not be economically sound. However, recent developments in micro technology might set new standards on software efficiency, which could result in a major shift from FORTRAN to other more efficient languages. Compilers which permit source code to be a combination of different programming languages (e.g. FORTRAN and C language) are already on the market. They will allow for restructuring of only those parts of huge programs which take longest to execute, which in many instances means changing only a couple of hundred lines of the program, as is the case with the OKA subroutines in WRMM.

Apart from that, a new generation of optimization algorithms (genetic algorithms [55], [56]) has a tremendous potential of improving efficiency of codes like WRMM. All of these algorithms are based on random generation of a number of feasibile solutions, comparing their objective functions and using only the best guesses for generation of the new population of feasibile solutions. The process is thus repeated as long as optimality of two subsequent iterations remains approximately the same. Further investigation is needed for comparison between Barr's algorithm and genetic algorithms. Also, generating a large number of feasibile solutions for circulatory capacitated networks must involve additional operations associated with maintaining nodal continuity and compliance with upper and lower bounds for all arcs, which might reduce the overall efficiency of genetic algorithm in this case.

6.6 FORTRAN Data Structure Limitations

All types of data that can be handled by FORTRAN programming language are restricted to a narrow choice of data organization, by which all relevant information is stored and retrieved by referencing corresponding locations in arrays. Other higher level programming languages offer a more effective data structure organization, which can greatly reduce the number of array referencing, thus improving the execution efficiency.

This section introduces a database model of Barr's algorithm which could be successfully utilized by PASCAL [54] programming language, as well as any other that incorporates the data structures described in the following.

6.6.1 Linked Lists vs Physically Ordered Lists

A linked list is a list of data items linked by "pointers" contained within the data. Terms "head" and "tail" indicate the pointers which point to the first and last data item in the list, respectively.

A one way linked list allows for search of the list in only one direction. This is a disadvantage to physically ordered lists, which allow sequential access of any data item. However, inserting and deleting data is much easier in the case of linked lists; it is only necessary to change values of same pointers, whereas in the case of physically ordered list inserting a data item in the middle of the list means that all the items below the inserting location have to be shifted down for one location.

To override the limitations of linked lists associated with their processing, circular and two way linked lists were designed. Circular linked lists differ from one way linked lists in the direction of the last pointer, which in the case of circular lists points to the first item. This allows for processing of all data starting from an arbitrary location. However, such processing is still directed in one way only. For example, if item i is processed and then items (i+1) and (i-1) are to be retrieved, the processing of (i+1) will be easy since the i^{th} pointer points to (i+1) location. However, to retrieve (i-1) all other data items have to be processed, starting from item (i+1).

To avoid this, linked lists with two pointers for each data item were designed, termed as two-way linked lists, which make possible processing of data in both directions. In a similar fashion, multiple linked lists can be installed to enable processing of data in different orders. They all require more storage than the physically ordered list. However, the advantages become obvious when data is restructured. Also, linked lists allow for logical organization of complex data structures, such as trees and networks.

6.6.2 PASCAL Data Structures

Along with some other higher level programming languages, PASCAL can handle data structures, such as sets, records or files. Rather than storing a set of logically linked data into arrays and accessing each member by indexing, these structures can contain whole arrays under one record, accompanied by built-in pointers which point to the location reserved for each data item individually. The advantages of this can be considerable. For example, in order to copy one array into another using FORTRAN, a loop must be set in the program, which retrieves an individual value of the first array and copies it into the appropriate value of the other. However, if the same array is stored as a record in PASCAL, all that is needed is to copy one record into another, i.e. one statement.

More importantly, this data organization is more flexible when individual data items are to be shifted from one location to another within an array, since it is enough to change the value of the corresponding pointers.

Even more exciting is the existence of sets as distinct data structures and the ability to incorporate directly some of the set operators such as union, intersection, or difference. One should recall that most of the computational effort exerted by the OKA is contained in isolation of various types of sets: set of label eligible and label ineligible arcs (the last one split into sets S and R), set of labelled and unlabeled nodes, etc. Keeping track of changes within these sets during execution is greatly simplified if the actual set operators are applied directly, which substantially eliminates the need for many "IF" statements and references contained in processing of the same sets organized as arrays in FORTRAN.

6.6.3 An Example of a Linked List Database Model for Barr's Algorithm

Consider an arc record as a set of all relevant information related to one pseudo arc, as depicted in fig 6.1.

ARC (RECORD) NUMBER	nc(b)	c(b)	KILTER STATUS	HEAD NODE POINTER

fig 6.1 ARC RECORD

For each pseudo arc b, corresponding net capacity and marginal cost are represented by numbers (either integer or real depending on the problem). The kilter status can be set as a boolean type variable, with only two values: "true" or "false", depending on whether an arc is in-kilter or not, respectively. Finally, the last position (head node) is a pointer, which has a fixed value that can't be changed. Noting that records can be stored within other records, consider a node record which contains corresponding arc records of outgoing pseudo arcs for that particular node:

NODE NUMBER	LABEL FLAGS		L.E. & S-R POINTERS		ARC RECORD POINTERS				
1	0	0	3	5	1	2	3	. . .	
2									
.									
.									
N	.	.	.	.	.	.	.	. . .	

fig 6.2 NODE RECORDS

The first indicator is the record (node) number. There are two label flag locations, instead of one as earlier, and this will be explained in the following. These two flags are to indicate whether the node is currently labelled or not. In addition to the label eligible pointer (which is equivalent of the value in array MIDL in the FORTRAN version) there is one more pointer which splits label ineligible arcs into sets S and R. One can recall that these sets had to be generated from scratch with each potential change, except for the case when two or more potential changes were subsequent. Finally, arc records are the same as described earlier, except that instead of arc number another indicator can be introduced, simply termed as a record pointer by which appropriate arc records are accessed (within one node record). Arc number in the previous sense was needed to provide information on the capacity, marginal cost and head node, which were all stored in arrays. Since arrays are eliminated in this data structure, arc number is no longer needed. Suppose, for example, that there are six outgoing pseudo arcs for node one, and that the current values in label eligible pointer and S-R pointer are 3 and 5, respectively. This indicates that first three arc records (indicated by the value of the appropriate pointers, rather than by their physical location) are records of corresponding label eligible arcs. Similarly, arcs indicated by pointers 4 and 5 would belong to set R. Values of these pointers would be initiated in the setup section and changed accordingly during program execution. Also initiated with zeros are both label flags for all nodes.

The following is a brief description of how Barr's algorithm would operate under this data organization: Isolate an out-of-kilter arc by starting from the first node record and accessing arc records, one by one, until an arc with "false" value of its kilter status is found. The algorithm now attempts the labelling procedure, which starts from the node indicated in the head node pointer of the starting arc and then either destination is reached (label of the starting node is assigned a non-zero value), or if there is no where else to go (i.e. sets P_i of all nodes labelled in the last cycle are empty) the algorithm attempts to extend the path by applying potential change procedure.

Labelling Procedure

Start from the node record which is indicated in the head node pointer of the out-of-kilter arc which is currently being brought in-kilter.

1. Access the value of label eligible pointer of this node. If this value is zero, there are no label eligible arcs so go to the potential change procedure section.
2. If there are label eligible arcs, repeat the following procedure for each of them (their number is indicated by the current value of the label eligible pointer): Access a label eligible arc record and label the appropriate node (indicated by the head node pointer of arc record). This label now consists of two parameters:
   a) arc record pointer (instead of arc number)
   b) node pointer (indicates previously labelled node in

the labelling procedure).

The same information is contained in the FORTRAN version of this program, but it was stored in two separate arrays (LABEL and NTAIL).

3. Once the labelling procedure results in breakthrough, it is necessary to isolate a chain of arcs that lie in the flow augmenting path. This is done much easier with the new data structure. Starting from the node which was labelled last (tail node of the out-of-kilter arc), access its label, which contains both arc pointer and the number of node labelled in the previous labelling cycle, whose label is accessed nest, and so forth until the starting node is reached. In doing so the appropriate arc record of each arc on the path is accessed and the values of their net capacities are compared, in order to isolate the one closest to zero. The whole retrace is then repeated along with changing net capacities of all arcs (and their mirrors) which lie in the path. During the second retrace subsets P, S and R are updated accordingly, which is done merely by changing values of the label eligible and S and R pointer, respectively.

After this, kilter status of the starting arc is examined again. If it is in kilter, the kilter status flag is set to "true" and the search continues for another out-of-kilter arc. Conversely, if the starting arc is still out of kilter another labelling procedure is initiated, with chances of resulting in breakthrough or potential change.

Potential Change Procedure

4. Access all labelled nodes, one by one, and for each of them access arcs which are currently in set S. Next determine if head nodes of such arcs are labelled; if not, store the relevant information on such arc as a member of external set S_{ext}. Each time a new member is added to this set, the appropriate value of its marginal cost is evaluated. This results in detection of the marginal cost being closest to zero while the set is being created. Paralelly with creating set S_{ext} and create set R_{ext} in a similar fashion.

The next step is change of marginal cost for all arcs in sets S_{ext} and R_{ext}, which are now readily available. After this set S_{ext} will lose at least one member, and one more node will be labelled, or the problem is declared infeasible.

Continuation of the labelling procedure might result in labelling some of the nodes which were not labelled before, and which constituted head nodes of certain arcs currently in S_{ext} and R_{ext}. If additional potential change is needed under such circumstances, sets S_{ext} and R_{ext} must be purged from currently nonconforming arcs, as well as extended by arcs in sets S and R of the newly labelled nodes. Instead of listing entire arrays containing arcs in sets S and R and conducting a number of checks on each arc to monitor the foregoing changes, PASCAL solves this problem with using standard set operators, such as union, intersection and difference.

Summary

The advantage of this type of data organization is that all relevant data is stored in a block form with built-in logical links which almost entirely eliminate the need for external referencing. The only situation when external referencing is required is the case of multiple potential changes (demonstrated in iteration for arc number 10 in the numerical example section). However, the amount of referencing in this section is greatly reduced by using the set data structure and set operators.

7. CONCLUSION

Computer modelling has become a major tool in many areas of science and management. There are two major factors which encourage this trend:

- advances in applied mathematics, primarily in the area of numerical analysis and mathematical programming
- advances in computer technology

The latest is especially of interest with respect to the recent developments in micro-computer technology. The major impact of these developments is that computers have become available to many people today, while two or three decades ago it was only a priviledged minority that had access to them. Also, the performance abilities of micro-computers have significantly improved over the past decade.

At the same time, many modelling concepts were developed and tested in various kinds of studies which involved application of computer models. As a result of this, certain problems were simplified and specialized algorithms were applied, which proved to be more efficient than the standard algorithms. This was a general trend in many areas which were studied using computer models.

Computer models applied in the area of water resources engineering have gone through the similar developments. Modelling concepts which were proved to be inefficient were abandoned. On the other hand, those concepts which showed more efficiency were used as the basis for further developments. In the case of surface water allocation models, one of the best approaches so far (known as "the zoning concept") was successfully applied in a number of models, including the Water Resources Management Model (WRMM) of Alberta Environment [39], developed in 1981. This concept is based on specifying a number of discrete zones for each modelled component (e.g. a number of flow zones for channel components, or a number of storage zones for storage components), and assigning a penalty to each zone. Each component has one ideal zone whose penalty is always set to zero. Apart from the ideal zone, all other zones have positive penalties associated with them. The values of the penalties represent priorities assigned to different components, i.e. when the natural supply is not sufficient to meet all the demands in a particular time interval, the first component to be affected is the one whose penalty below the ideal zone is the closest to zero.

A special type of linear programming solution technique, known as the "Out-of Kilter" algorithm (OKA) [36] was applied in this model, due to its superior efficiency over other linear programming techniques that could have been applied at that time [49]. However, later advances of this solution technique by Barr et al. [52] prompted the replacement of the original OKA code. The purpose of this study was to implement the Barr's code in WRMM, to evaluate the impacts of the new code on WRMM and to indicate directions for possible improvements in the future.

A new code based on Barr's algorithm was successfully implemented in WRMM and the overall execution efficiency was improved by two to three times on average, but this is only the case for large networks. This limitation was to be expected, since the nature of Barr's improvements is such that the execution efficiency increases exponentially with the increase in the size of the network.

Apart from the improvement in execution efficiency, this study points out some other aspects which could be incorporated in future developments of WRMM or similar models:

1) the advantage of using INTEGER variables instead of REAL for models which simulate long time periods
2) the advantage of using more efficient data structure other than arrays
3) further investigation of Barr's code and its possible improvemens

Once developed, the latest improvements could be compared to other optimization techniques, like the dynamic programming or genetic algorithms. Both of these approaches require solution of certain technical problems related to their successfull application in WRMM. Having solved these problems, it would be possible to compare a number of different optimization algorithms and evaluate their overall performance and impacts.

REFERENCES

1) Alvin S. Goodman, "Principles of Water Resources Planning", Prentice Hall Inc., New Jersey, 1984.

2) Sherman L.K., "Streamflow from Rainfall by Unit-Graph Method", Eng. News Rec., vol 108, p. 501-505, 1932.

3) Thomas H.A., Fiering M.B., "Mathmatical Synthesis of Streamflow Sequences for the Analysis of River Basins by Simulation", Design of Water resources Systems, Harvard University Press, chapter 12, p. 459, 1962.

4) Roesner L.A., Yevjevich V.M., "Mathematical Models for Time Series of Monthly Precipitation and Monthly Runoff", Colo. St. Univ., Fort Collins, Hydrology Papers, No. 15.

5) Crawford N.H., Linsley R.K., "Digital Simulation in Hydrology: The Stanford Watershed Model IV", Tech. Rept. No 39.,Dep. of Civil Eng., Stanford University, 1966.

6) U.S. Army Corp. of Engineers, "Program Description and User Manual for SAARR Model Streamflow Synthesis and Reservoir Regulation", Program 724-K5-G0010, Dec. 1972.

7) Biswas A.K., "Systems Approach to Water Management", McGraw Hill Inc., 1976.

8) Hall W.A., Dracup J.A., "Water Resources Systems Engineering", McGraw Hill Inc., 1970.

9) Buras N., "Scientific Allocation of Water Resources", American Elsevier Publishing Co. Inc., 1972.

10) Goldberg D.E., Thomas A.L., "Genetic Algorithms: A bibliography 1962-1986", TCGA Report No. 86001., Dep. of

Eng. Mechanics, Univ. of Alabama, 1986.

11) Beard L.R., "Functional Evaluation of a Water Resources System", presented at Int. Conference on Water for Peace, Washington D.C., May 23-31, 1967.

12) Deneufville R., Marcs D.H., "Systems Planning and Design: Case Studies in Modelling, Optimization and Evaluation", Prentice Hall Inc., New Jersey, 1974.

13) Major D.D., "Benefit-Cost Ratios for Projects in Multiple Objective Investment Programs",Water Res. Research, vol 5, No 6, 1961.

14) Roy B., "Problems and Methods with Multiple Objective Function", Math. Programming, vol 1, 1971.

15) Thomas H.A., Watermeyer P., "Mathematical Models: A Stochastic Sequential Approach in Design of Water Resource Systems", chap 14, p. 540-561, Harvard University Press, Cambridge, Mass., 1962.

16) Little J.D.C., "The Use of Storage Water in a Hydro Electric System", Operations Research, vol 3, p. 187, 1955.

17) Buras N., "Conjunctive Operation of Dams and Aquifers", J. Hydrol. Div. A.S.C.E., 89(HY6), 111, 1963.

18) Hall W.A., "Optimum Design of a Multy-Purpose Reservoir", J. Hydr. Div., A.S.C.E., 90(HY4), 141, 1964.

19) Young G.K., "Techniques for Finding Reservoir Operating Rules", Ph.D. thesis, Harvard Univ., Cambridge, Mass., 1966.

20) Young G.K., "Finding Reservoir Operating Rules", J.

Hydr. Div., A.S.C.E., 93(HY6), 297, 1967.

21) Harboe R.C., Mobasheri F., Yeh W.G., "Optimal Policy for Reservoir Operation", J. Hydr. Div., A.S.C.E., 96(HY11) 2297-2308, 1970.

22) Parikh, S.C., "Linear Dynamic Decomposition Programming of Optimal Long-Range Multiple Multipurpose Reservoir System', Oper. Res. Center, Univ. of Calif., Berkeley, Calif. 1966.

23) Buras N., "The Optimization of Large Scale Water resources Systems, 1, Conceptual Framework, p. 1-34, Univ. of Calif. Press, Berkeley, Calif., 1965.

24) Hall W.A., Tauxe G.W., Yeh W.G., "An Alternative Procedure for for the Optimization of Operations for Planning with Multiple River, Multiple Purpose System", Water Res. Resarch, 5(6), 1367-1372, 1969.

25) Gablinger M., Loucks D. P., "Markov Models for Flow Regulation", J. Hydr. Div., A.S.C.E., 96(HY1) 165-181, 1970.

26) Roefs T.G., Bodin L.D., "Multireservoir Operation Studies", Water Res. Research, 6(2), 410-420, 1970.

27) Revelle C., Kirby W., " Linear Decision rule in Reservoir Management and Design, 2, Performance Optimization", Water Res. Research, 6(4), 1033-1044, 1970.

28) Charnes A., Cooper W., Symonds G.W., "Cost Horizons and Certainty Equivalents: An Approach to Stochastic Programming of Heating Oil", Management Sci., 4(3),

235-263, 1958.

29) Eastman J., Revelle C., " The Linear Decision rule in Reservoir Management and Design, 3, Direct Capacity Determination and Interseson Constraints", Water Res. Research, 9(1), 29-42, 1973.

30) Leclerc G., Marcs D.H. "Determination of the Discharge Policy for Existing Reservoir Networks under Differing Objectives", Water Res. Research, 9(5), 1155-1165, 1973.

31) Nayak S.C., Arora S.R., "Optimal Capacities for a Multireservoir System using the Linear Decision Rule", Water Res. Research, 7(3), 485-498, 1971.

32) Loucks D.P., Dorfman P.J., "An Evaluation of Some Linear Decision Rules in Chance Constrained Models for Reservoir Planning and Operation", Water Res. Research, 11(6), 1975.

33) Eisel L.M., "Chance Constrained Reservior Model", Water Res. Research, 8(2), 339-347, 1972.

34) Frederich A.J., Beard L.R., "System Symulation for Integrated Use of Hydroelectric and Thermal Power Generation", Tech. Pap. 33, Hydrol. Eng. Center, U.S. Army Corp. of Engineers, Davis, Calif. 1972.

35) Tedrow A.C., Liu S.C., Halton D.B., Hiney R.A., "The Use of Systems Analysis in the Development of Water Resources Management Plans for New York State", Report to U.S. Dep. of Interior, Office of Water Res. Research, Dep. of Env. Conserv., Div. of Water R., Albany, N.Y., 1970.

36) D. R. Fulkerson, "An Out-of-Kilter Method for Minimal Cost Flow Problems", Journal of the Society for Industrial and Applied Mathematics, vol 9, no 1, 1961.

37) D. R. Fulkerson, L. R. Ford, "Flows in Networks", Princeton University Press, Princeton, N.J., 1962.

38) ~~Texas~~ Water Development Board, "Economic Optimization and Simulation Techinques for Management of Regional Water Resources Systems - River Basin Simulation Model SIMYLD-II Description", 1972.

39) Alberta Environment, Planning Division, "Water Resources Management Model, Computer Program Description", April 1987.

40) Acres Consulting Services, "The Trent Basin Analysis of the System", Report to the Dep. of Indian and Northern Affairs, Niagara Falls, 1973.

41) Nanuk Engineering & Development Ltd., "The Water Balance Simulation Model - Programmers Manual", April 1982.

42) O. T. Sigvaldason, "A Simulation Model for Operating a Multipurpose, Multireservoir System," Water Resources Research, vol. 12, no. 2, April 1976.

43) M. S. Bazaraá, J. J. Jarvis, "Linear Programming and Network Flows", John Wiley & Sons Inc., 1977.

44) J. Von Neuman, "On a Maximization Problem", Institute for Advanced Studies, Princeton, N.J. 1947.

45) H. W. Khun, A. W. Tucker, "Linear Inequalities and Related Systems", Anals of Mathematical Study no. 38, Princeton University Press, Princeton, N.J., 1956.

46) A. W. Tucker, "A Combinatorial Theory Underlying Linear Programs", McGraw Hill Book Co., New York, 1963.

47) W. S. Jewel, "Complex: A Complementary Slackness, Out-of-Kilter Algorithm for Linear Programming", ORC 67-6, University of California, Berkeley, Calif., 1967.

48) H. W. Khun, A. W. Tucker, "Nonlinear Programming", in J. Neuman (ed.), "Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability", University of California Press, Berkeley, Calif. 1950.

49) G. B. Dantzig, "Linear Programming and Extentions", Princeton University Press, Princeton, N.J., 1966.

50) M. Simonnard, "Linear Programming", Prentice Hall, Englewood Cliffs, N.J., 1966.

51) E. P. Durbin, D. M. Kroenke, "The Out-of-Kilter Algorithm, A Primer", RM 5472-R, p. 22, Rand Corporation, Santa Monica, Calif., 1967.

52) R. S. Barr, F. Glover, D. Klingman, "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes", Mathematical Programming 7, p. 60-86, North Holland Publishing Company, 1974.

53) M. Klein, "A Primal Method for Minimal Cost Flow Problem with Applications to Assignment and Transportation Problems", Management Science, vol 14, no 3, 1967.

54) Tenenbaum A., "Data Structures Using PASCAL", Englewood Cliffs, N.J., 1981.

53) Goldberg D.E., Samtoni M.P., "Engineering Optimization via Genetic Algorithms", Proceedings of the 9th conf. on

electronic computation, A.S.C.E., N.Y., 1986.

53) Grefenstette J.J., (ed.) "Proceedings on an International Conference on Genetic Algorithms and Their Applications", Charnegie Mellon Univ., Pittsburg, P.A., 1985.

APPENDIX I

This appendix contains the alphabetical listing of all the variables which are of interest regarding the KILTER subroutines. The variables are listed in two groups, depending on whether they are created in the main program or in the subroutines. All of the variables passed from the main program except INFES, MARCS2, MND1, R and Z existed in the WRMM prior to the introduction of the new KILTER subroutines.

VARIABLES CREATED IN THE MAIN PROGRAM

Variable	*Definition*
ACOST	penalty incurred per unit of flow in an arc
AFLOW	amount of flow in an arc
ARCUB	upper bound of an arc
ARCLB	lower bound of an arc
INFES	infeasibility flag
MARCS	total number of arcs in the network
MARCS2	total number of pseudo arcs in the pseudo network
MND	number of nodes in the network
MND1	number of nodes in the network increased by one
NHEAD	array of head nodes of all pseudo arcs
NTAIL	array of tail nodes of all pseudo arcs
NCHA	number of arcs created to represent one component of the system
NSETS	number of arc sets in the system
NLOWST	the lowest arc set number associated

	with a node
NHIST	the highest arc set number associated with a node
SETMAP	array of arc set numbers associated with each physical channel
R	Double precision constant set at 10^{-7}
Z	Double precision constant set at 10^{3}

VARIABLES CREATED IN THE SUBROUTINES

Variable	*Definition*
FIP	flow increase potential of an arc that lies in the flow augmenting path
FLWCHG	amount of flow change exerted on all arcs in the flow augmenting path
IAC	arc number in subroutines PINQ and QINP
INLOC	array of labelled nodes
IM	number of the main pseudo arc
IMIR	number of the mirror pseudo arc
IT	head node of the pseudo arc being currently brought in-kilter
IS	tail node of the pseudo arc being currently brought in-kilter
JARC	temp. storage for a pseudo arc number
JL	temporary storage for a pseudo arc number in the multiple potential change section
JNEXT	temp. storage for the head node of a pseudo arc
JNODE	a multipurpose array used for temporary

	storage in subroutines PSARCS, ARRAYS and KILTER.
KARCS	variable which indicates if the labelling procedure starts from the head node or tail node of the starting arc
LABEL	array of labels; contains arc numbers for labelled nodes and zeros for unlabeled nodes
MIDL	array which points to the last position of a label eligible pseudo arcs for a node
NBRK	the flag indicating if breakthrough occurred as a result of potential change
NDARCS	array of outgoing pseudo arcs for all nodes
ND1	temporary storage for pseudo arc number
NEND	temporary storage for the last position of an arc in sets P_i or Q_i
NEWNOD	head node of an arc in set P_i
NLBA	arc number of the label eligible arc after one potential change
NLN	number of labelled nodes
NLNCRS	number of labelled nodes whose arcs in set Q_i are checked for membership in sets S or R
NMA	number of mirror arc

NNLS	number of labelled nodes whose set P_i have been labelled
NNLS1	storage for NNLS in case of multiple use of the labelling procedure
NODE	array which points the first position of an outgoing pseudo arcs for each node
NPA	number of preceding arc
NR	number of arcs in set R
NRLPCH	number of arcs in set R during the last potential change
NS	numnber of arcs in set S
NSN	temporary storage for a node number in retracing of the flow augmenting path
NSN1	dummy argument in subroutine PINQ, contains a node number
NSLPCH	number of arcs in set S from the last potential change.
NSETS	number of arc sets in the network
NSTART	temporary storage for the first position of an arc in set Q_i or P_i
PACCHG	amount of marginal cost change
PACOST	marginal cost of a pseudo arc
PACOS1	temporary storage for PACOST

PANC	net capacity of a pseudo arc
PANC1	temporary storage for PANC
VNODE	array used for temporary storage of all net capacities of arcs (or their mirrors) which lie in the flow augmenting path

APPENDIX II

This appendix contains the following source code listings:

1) The WRMM main program listing
2) Kilter subroutines of the adopted version
3) Kilter subroutines of the experimental version

The main program is given with the adopted version of the KILTER subroutines. The other version of the main program differs only slightly in the following:

1) Arrays MIDL and NODE are called MIDP and NODP and their dimension is 642 instead of 102. They are associated with each arc set in the network, rather than each node.
2) Additional arrays NSETA is needed to identify arc set number for each arc in the network. This array has a dimension of 1000.
3) Additional variable NSTS1(=NSETS+1) is initialized (before calling the subroutine PSARCS) and passed to the kilter subroutines, along with NSETS.

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:38              PAGE:     1
REQUESTED OPTIONS (EXECUTE): NOTRMFLG,SOURCE,IL(DIM)
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE   TERM   OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
                    NOSYM NORENT   SDUMP AUTODBL(NONE)   NOSXM   IL
                      OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)     CHARLEN(500)
            *....*...1.........2.........3.........4.........5.........6.........7.*.......8
            C    *
            C    *    NAME: W. R. M. M. MAIN PROGRAM
            C    *
            C    *    FUNCTION: THIS FORTRAN PROGRAM IS THE
            C    *              MAIN CALCULATION PROCEDURE FOR DETERMINING
            C    *              THE MINIMAL PENALTY FLOWS IN A GENERAL WATER
            C    *              BALANCE NETWORK.
            C    *
            C    *    FILES USED:     SCF SUPPLIED BY USER, READ FROM FORTRAN
            C    *                            UNIT NUMBER KSCF.
            C    *                    CDF       ARCHIVAL COMPONENT DATA
            C    *                    DEBUG     THE OUTPUT FILE OF ERROR INFORMATION.
            C    *                              UNIT NUMBER KDEBUG.
            C    *                    PRINT     THE OUTPUT FILE OF NORMAL OUTPUT. UNI
            C    *                              NUMBER KPRINT.
            C    *
            C    *         IDENT1     READS THE SCF SUB-FILE $IDENT
            C    *         SMCN1      PERFORMS THE FIRST PASS READ OF THE SCF SUB-FIL
            C    *                    $SIMCON
            C    *         PHSYS1     PERFORMS THE FIRST PASS READ OF THE SCF SUB-FIL
            C    *                    $PHYSYS
            C    *         PNSYS1     PERFORMS THE FIRST PASS READ OF THE SCF SUB-FIL
            C    *                    $PENSYS
            C    *         WTDM1      PERFORMS THE FIRST PASS READ OF THE SCF SUB-FIL
            C    *                    $WATDEM
            C    *         WTSP1      PERFORMS THE FIRST PASS READ OF THE SCF SUB-FIL
            C    *                    $WATSUP
            C    *
            C    *         ALL THE FIRST PASS READ SUBROUTINES CHECK FOR INCORRECT
            C    *         DECK STRUCTURE AND MISSING OR INCORRECT RANGE DATA, AND
            C    *         CAUSE PROCESSING TO CEASE IMMEDIATELY IF A PROBLEM IS
            C    *         DETECTED.
            C    *         THESE SUBROUTINES ALL SHARE A COMMON NAMING CONVENTION,
            C    *         WHICH IS DESCRIBED AS FOLLOWS:
            C    *
            C    *         THE FIRST LETTER DENOTES THE NATURE OF THE INFORMATION
            C    *               I     A SCRATCH OR WORKING INDEX
            C    *               J     A SCRATCH OR WORKING INTEGER VARIABLE
            C    *               K     A PERMANENT (GLOBAL) INTEGER VARIABLE
            C    *               L     THE MINIMUM (LEAST) NUMBER OF ---
            C    *               M     THE MAXIMUM (MOST)  NUMBER OF ---
            C    *               N     THE NUMBER OF ---
            C    *
            C    *         THE SECOND AND THIRD LETTERS INDICATE THE "SUBJECT"
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:38      NAME: MAIN    PAGE:    2
                     *....*...1.........2.........3.........4.........5.........6.........7.*.......8
                     C        *           TM   TIME OR TIMES
                     C        *           NB   NUMBER OF ---
                     C        *           ZA   NUMBER OF ZONES ABOVE THE RULE CURVE
                     C        *           ZB   NUMBER OF ZONES BELOW THE RULE CURVE
                     C        *           ZN   NUMBER OF ZONES
                     C        *
                     C        *     THE FOURTH AND FIFTH LETTERS INDICATE THE GENERATING PGM
                     C        *           SM   SMCN1
                     C        *           PH   PHSYS1
                     C        *           PN   PNSYS1
                     C        *           WD   WTDM1
                     C        *           WS   WTSP1
                     C        *
                     C        *     THE SIXTH LETTER ENCODES THE SUB-FILE ELEMENT INVOLVED
                     C        *           R    RESERVOIR
                     C        *           I    IRRIGATION
                     C        *           H    HYDRO
                     C        *           J    JUNCTION (NODE)
                     C        *           D    DIVERSION CHANNEL
                     C        *           N    NATURAL CHANNEL
                     C        *           M    MAJOR WITHDRAWAL
                     C        *           W    MINOR WITHDRAWAL
                     C        *           A    APPORTIONMENT
                     C        *           C    CONTROL STRUCTURE
                     C        *           E    EVAPORATION
                     C        *           P    PRECIPITATION
                     C        *
                     C        *     THE REMAINING PASSED PARAMETERS ARE:
                     C        *          JWRD A CHARACTER VARIABLE CONTAINING COLUMNS
                     C        *               2-5 OF THE $ PREFIX CARD JUST READ IN
                     C        *               THE SUBROUTINE
                     C        *           IER  A TERMINATION STATUS FLAG
                     C        *                 -1 IF EOF WAS ENCOUNTERED CORRECTLY
                     C        *                 0  IF SUB-FILE ENDED AT ANOTHER
                     C        *                    $ PREFIX CARD
                     C        *                 1  IF AN ERROR CONDITION AROSE
                     C        *
                     C        *
                     C        ***************************************************************
                     C
ISN        1                  INTEGER*4 NDMAP(102,9),NSSNA(102),NSBNA(102),NRTNA(102)
ISN        2                  INTEGER*4 NHIST(102),NLOWST(102),NMOOUT(60),KCYCLA(20)
ISN        3                  INTEGER*4 CHMAP(321,5),NCHA(321),NCHAB(321),TYPCOD(321)
ISN        4                  INTEGER*4 SETMAP(642),NODARC(642),NEXNOD(642),KINTRA(60)
ISN        5                  INTEGER*4 NHEAD(2000),NTAIL(2000),ZN(60,321,2),NVROUT(60)
ISN        6                  INTEGER*4 INTNBR(1002),MECMHN(20),STNODE
ISN        7                  INTEGER*4 NDARCS(2000),XKMAP(1002,4),JNODE(2000),NODE(103)
ISN        8                  INTEGER*4 NDOUT(70),LKOUT(70),JMET(6),NDVOUT(60),KEYPDF(321)
```

```
                    *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN        9          INTEGER*4 H$,HIDENT,HSIMCO,HPHYSV,HPENSY,HWATDE,HWATSU
ISN       10          INTEGER*4 KEYCHA(20),KEYCHD(40),KEYCHH(20),KEYCHI(25)
ISN       11          INTEGER*4 KEYCHM(20),KEYCHN(75),KEYCHR(30),KVRCHI(20,25)
ISN       12          INTEGER*4 KVRCHM(20,20),KEYCHC(30),KEY(321),KVLTMA(20)
ISN       13          INTEGER*4 KTPPHC(30),KRFPHC(30),KTPPHH(2,20)
ISN       14          INTEGER*4 KMTPHR(20,30),KEYPCR(30),KEYPCI(25),KEYPCM(20)
ISN       15          INTEGER*4 KEYPCH(20),KEYPCN(75),KEYPCA(20),KEYPCD(40)
ISN       16          INTEGER*4 KTPWSA(20),KEYCHW(50),KEYCHF(80),KMTPDF(20)
ISN       17          INTEGER*4 DUSWCH(26),LKH1,LKH2,KTM(60),MIDL(102),LABEL(102)
ISN       18          INTEGER*4 PACOST(2000),INLOC(102),PACCHG,PACOS1
ISN       19          REAL XSCR4(250)
ISN       20          REAL*8 REFNAM,STUDY(10),SYSTEM(10),RUN(10),USER(2),DATE
ISN       21          REAL*8 REMARK(7),TITLE(25),OUTDAT(60,321),TMCYCA(60),FTM(60)
ISN       22          REAL*8 XSTR(6),XMET(6),ARCLB(1000),ARCUB(1000),VNODE(102)
ISN       23          REAL*8 AFLOW(1000),ACOST(1000),USERNM(321),OTFLLK(70),OTFLND(70)
ISN       24          REAL*8 VLMINR(30),VLINCR(30),ELVLR(20,30),EWTPHR(20,30)
ISN       25          REAL*8 AREAI(25),RRTNI(20,25),PRTNM(20,20),GNMET(30),ZELEVS(10)
ISN       26          REAL*8 RHDPHH(20),QMAXH(20),QPPHH(20),ELFLHH(20,20),FLMNHH(20)
ISN       27          REAL*8 FLICHH(20),ELFLTH(20,20),FLMNTH(20),FLICTH(20),HDLPHH(20)
ISN       28          REAL*8 RNOFFA(20),FLMINA(20),VLMINA(20),ZFLOWN(10),TMINT(60)
ISN       29          REAL*8 FLELC(20,30),ELMNC(30),ELICC(30),TRTNM(20),VLTGA(60,20)
ISN       30          REAL*8 RCTMR(60,30),ZELTMR(10,60,30),ZELPUI(10),TMINTA(60)
ISN       31          REAL*8 RCTMI(30,25),ZFLTMI(10,30,25),PANC(2000),FLWCHG,FIP
ISN       32          REAL*8 ZFLPUM(10,20),ZPDPUH(10,20),RCTMN(60,75),VLACHA(20)
ISN       33          REAL*8 ZFLTMN(10,60,75),ZFLPUA(10,20),RCTMD(60,40),TMELA
ISN       34          REAL*8 ZFLTMD(3,60,40),WDTMW(60,20),PDTMH(60,20),XSCR(60,100)
ISN       35          REAL*8 WDTMI(60,25),RFFTMI(60,25),DEFTMI(60,25),PDFNAM(2,250)
ISN       36          REAL*8 RFCTMI(60,25),ZFLOWD(3),ZMXVLD(40),TMXVLD(40),FWDJ
ISN       37          REAL*8 WDTMM(60,20),WSTMF(60,80),FFLTMA(60,20),ZMXFLD(40)
ISN       38          REAL*8 HDROP(20),FWOLD(30),OUTFIL(60,321),FHOLD(20),FHASM(20)
ISN       39          REAL*8 TIDEAL(60,321),OUTEVP(60,30),CTLFX(30),Z
ISN       40          REAL*8 XTM(60),EVPRTM(60,20),DIVHD(40),TMOFST,TMREL,TMCYC,TMCYMX
ISN       41          REAL*8 CTLHD(30),FXOLD(20),HDOLD(20),TMV,STOARC(321),APPVOL(20)
ISN       42          COMMON /DIMDAT/MNBPHR,MPTPHR,MNBPHM,MRTPHM,NRTPHM,MNBPHH,MNWPHH,
                     +MTWPHH,MNBPHI,MRTPHI,NRTPHI,MNBPHN,MNBPHA,MNBPHD,MNBPHJ,MNBPHC,
                     +MMTPHR,MPTPHC,MNBPNI,MNBPNR,NZAPNR,NZBPNR,MPCPNI,MZNPNI,NZNPNI,
                     +MNBPNM,MZNPNM,MPCPNM,NZNPNM,MNBPNH,MZNPNH,MPCPNH,NZNPNH,MNBPNN,
                     +MZAPNN,MZBPNN,HZAPNN,NZBPNN,MNBPNA,MZNPNA,MPCPNA,NZNPNA,MNBPND,
                     +MPCPND,MPCPNN,MTMPNN,MTMPND,MNBWDW,MNBWDH,MNBWDI,MNBWDM,MNBWSR,
                     +MNBWSJ,MNBWSA,MTMWSA,MNBWSE,MNBWSP,MUJWSJ,MZNPNR,MZNPNN,MTMPNR,
                     +MTMWDW,MZAPNR,MZBPNR,MNBMET,MTMPNI
ISN       43          COMMON /TIMBLK/LID,LIM,LIV,LIHR,LIMN,LISE
ISN       44          COMMON /IOBLK/KSCF,KPDF,KPRINT,KDEBUG,KWRITE
ISN       45          COMMON /DEBGBK/DUSWCH,LKH1,LKH2,MOUTAL
                    C...INITIALIZATION SECTION
ISN       46          DATA ZN/38520*0/
ISN       47          DATA STOARC/321*0./
ISN       48          DATA NDMAP,NSSNA,NSBNA,NRTNA,NHIST,NLOWST/1428*0/
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:38      NAME: MAIN    PAGE:    4
                       *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN       49          DATA CHMAP,NCHA,NCHAB/2247*0/
ISN       50          DATA INTNBR,MNDNBR,MCHNBR,MARCS,NSETS/1006*0/
ISN       51          DATA SETMAP,NODARC,NEXNOD/1926*0/
ISN       52          DATA NHEAD,NTAIL,KMTPHR/4600*0/
ISN       53          DATA XKMAP,LABEL/4110*0/
ISN       54          DATA ACOST,AFLOW,ARCUB,ARCLB,RCTMR,ZELTMR/23800*0./
ISN       55          DATA H$,HIDENT,HSIMCO,HPHYSY,HPENSY,HWATDE,HWATSU/1H$,4HIDEN,
                     +4HSIMC,4HPHYS,4HPENS,4HWATD,4HWATS/
ISN       56          Z=1.D03
ISN       57          R=1.D-07
ISN       58          INFES=0
ISN       59          INFC=0
ISN       60          PDFFLG=0
ISN       61          MPDFNM=1
ISN       62          NDMAP(101,1)=1001
ISN       63          NDMAP(102,1)=1002
ISN       64          INTNBR(1001)=101
ISN       65          INTNBR(1002)=102
                C...INPUT/OUTPUT DEVICE UNIT NUMBER
ISN       66          KSCF=5
ISN       67          KDEBUG=6
ISN       68          KPRINT=6
ISN       69          KPDF=7
ISN       70          KWRITE=3
ISN       71          MCYCLA=0
ISN       72          MINTRA=0
                C...BEGIN FIRST PASS READ
ISN       73          READ(KSCF,1000)JCHR,JWRD
ISN       74    1000  FORMAT(A1,A4)
ISN       75          IF(JCHR .EQ. H$)GO TO 100
ISN       76          WRITE(KDEBUG,1001)
ISN       77    1001  FORMAT(1X,47HMAINWB: SCF DOES NOT BEGIN WITH A $ PREFIX CARD)
ISN       78          STOP
ISN       79    100   IF(JWRD .NE. HIDENT)GO TO 101
                C...SUB-FILE IDENT
ISN       80          CALL IDENT1(KDEBUG,KSCF,IER,JWRD)
ISN       81          GO TO 107
ISN       82    101   IF(JWRD .NE. HSIMCO)GO TO 102
                C...SUB-FILE SIMCON
ISN       83          CALL SMCN1(KSCF,KDEBUG,MINTRV,MCYCLE,MCYCLA,MOUTND,MOUTLK,JWRD,
                     +IER,MINTRA,MOUTEV,MOUTAL,NXKAPP)
ISN       84          GO TO 107
ISN       85    102   IF (JWRD .NE. HPHYSY)GO TO 103
                C...SUB-FILE PHYSYS
ISN       86          CALL PHSYS1(KSCF,KDEBUG,NDMAP,NSSNA,NSBNA,NRTNA,INTNBR,CHMAP,NCHA,
                     +NCHAB,JWRD,IER,MNDNBR,MCHNBR,XKMAP,MCHNXK)
                C...CHECK APPTCHL CONTROL CARDS
ISN       87          IF(IER.GT.0) GO TO 107
```

```
LEVEL 1.4.1 (MAY 1985)           VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:38      NAME: MAIN     PAGE:    5
                   *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN      88          IF((NXKAPP.EQ.MNBPHA).OR.(NXKAPP.GE.1.AND.MNBPHA.GE.1))
                    +GO TO 110
ISN      89          IF(NXKAPP.EQ.0) WRITE(KDEBUG,1002)
ISN      91          IF(NXKAPP.NE.0) WRITE(KDEBUG,1004)
ISN      93          STOP
ISN      94    1002  FORMAT(1X,'MAINWB: NO APPTPER AND ADJINT IN SIMCON FOR APPTCHL')
ISN      95    1004  FORMAT(1X,'MAINWB: APPTPER AND ADJINT SPECIFIED BUT NO APPTCHL',
                    +' IN PHYSVS')
ISN      96    110   CALL XKCHK(XKMAP,MCHNXK,KDEBUG,1)
ISN      97          GO TO 107
ISN      98    103   IF(JWRD .NE. HPENSY)GO TO 104
               C...SUB-FILE PENSYS
ISN      99          CALL PNSYS1(KSCF,KDEBUG,NDMAP,NSSNA,NSBNA,NRTNA,INTNBR,CHMAP,NCHA,
                    +NCHAB,JWRD,IER,MNDNBR,MCHNBR,XKMAP)
ISN     100          MZNPNR=MZAPNR+MZBPNR+2
ISN     101          MZNPNN=MZAPNN+MZBPNN+1
ISN     102          GO TO 107
ISN     103    104   IF(JWRD .NE. HWATDE)GO TO 105
               C...SUB-FILE WATDEM
ISN     104          CALL WTDM1(KSCF,KDEBUG,MINTRV,NDMAP,NSSNA,NSBNA,NRTNA,INTNBR,
                    +CHMAP,NCHA,NCHAB,JWRD,IER,MNDNBR,MCHNBR,MCYCLE,PDFFLG,XKMAP)
ISN     105          GO TO 107
ISN     106    105   IF(JWRD .NE. HWATSU)GO TO 106
               C...SUB-FILE WATSUP
ISN     107          CALL WTSP1(KSCF,KDEBUG,MINTRV,MINTRA,NDMAP,NSSNA,NSBNA,NRTNA,
                    +INTNBR,CHMAP,NCHA,NCHAB,JWRD,IER,MNDNBR,MCHNBR,PDFFLG,XKMAP)
ISN     108          MNBMET=MNBWSE+MNBWSP
ISN     109          GO TO 107
               C...INCORRECT SUB-FILE SETUP
ISN     110    106   WRITE(KDEBUG,1003)
ISN     111    1003  FORMAT(1X,46HMAINWB: INCORRECT NAME FOLLOWS $ PREFIX IN SCF)
ISN     112          WRITE(KDEBUG,999)JWRD
ISN     113          STOP
ISN     114    107   IF(IER)109,100,108
ISN     115    108   WRITE(KDEBUG,1005)
ISN     116    1005  FORMAT(1X,44HMAINWB: ERROR CONDITION RAISED BY SUBROUTINE)
ISN     117          WRITE(KDEBUG,999)JWRD
ISN     118    999   FORMAT(1X,'COLUMNS 2-5 OF THE LAST CARD READ CONTAINED ',A4)
ISN     119          STOP
ISN     120    109   CALL XKCHK(XKMAP,MCHNXK,KDEBUG,0)
ISN     121          REWIND KSCF
               C...FILL ARRAYS FOR SSN AND SBN. NODE NUMBER OF SBN BECOMES
               C...MNDNBR+1, OF SSN BECOMES MNDNBR+2.
               C...ARC FROM SBN TO SSN
ISN     122          NSSNA(MNDNBR+1)=2
ISN     123          NSBNA(MNDNBR+2)=2
ISN     124          CHMAP(MCHNBR+1,1)=1000
ISN     125          CHMAP(MCHNBR+1,2)=MNDNBR+1
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:38      NAME: MAIN    PAGE:    6
                  *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN      126            CHMAP(MCHNBR+1,3)=MNDNBR+2
ISN      127            NCHA(MCHNBR+1)=2
                  C...RENUMBER SBN AND SSN
ISN      128            NDMAP(MNDNBR+1,1)=1001
ISN      129            NDMAP(MNDNBR+2,1)=1002
ISN      130            INTNBR(1001)=MNDNBR+1
ISN      131            INTNBR(1002)=MNDNBR+2
ISN      132            DO 111 I=1,MCHNBR
ISN      133            IF(CHMAP(I,3) .EQ. 101)CHMAP(I,3)=MNDNBR+1
ISN      135            IF(CHMAP(I,2) .EQ. 102)CHMAP(I,2)=MNDNBR+2
ISN      137      111   CONTINUE
                  C...CALCULATE OKA ARRAY DIMENSIONS
ISN      138            MND=MNDNBR+2
ISN      139            MCH=MCHNBR+1
ISN      140            JCHA=0
ISN      141            DO 112 I=1,MCH
ISN      142      112   JCHA=JCHA+NCHA(I)
ISN      143            MARCS=JCHA
ISN      144            MINFLN=0
ISN      145            DO 113 I=1,MNDNBR
ISN      146            IF(NDMAP(I,7) .GT. 0)MINFLN=MINFLN+1
ISN      148      113   CONTINUE
ISN      149            MJCNDS=0
ISN      150            DO 114 I=1,MNDNBR
ISN      151            IF(NSBNA(I) .EQ. 0)MJCNDS=MJCNDS+1
ISN      153      114   CONTINUE
ISN      154            NSETS=2*MCH
                  C...RECONSTRUCT THE MODELLED SYSTEM TO CONFORM TO THE OKA REQUIREMENT
ISN      155            CALL TOPOL(KSCF,KDEBUG,MNDNBR,MCHNBR,NODARC,NEXNOD,SETMAP,NSETS,
                       +CHMAP,NCHA,NCHAB,NHEAD,NTAIL,MARCS,NHIST,NLOWST,MND,MCH)
                  C...BEGIN SECOND PASS READ, PENALTY AND BOUND LOADS.
ISN      156      135   READ(KSCF,1000)JCHR,JWRD
ISN      157            WRITE(KWRITE) JCHR,JWRD
ISN      158            WRITE (KPRINT,7000) JCHR,JWRD
ISN      159      7000  FORMAT(1H1,A1,A4)
ISN      160      115   IF(JWRD .NE. HIDENT)GO TO 116
                  C...SUB-FILE IDENT
ISN      161            CALL IDENT2(REFNAM,STUDY,SYSTEM,RUN,USER,DATE,REMARK,
                       +TITLE,JWRD,IER)
ISN      162            GO TO 122
ISN      163      116   IF(JWRD .NE. HSIMCO)GO TO 117
                  C...SUB-FILE SIMCON
ISN      164            CALL SMCN2(MINTRV,MCYCLE,MCYCLA,MOUTND,MOUTLK,JWRD,
                       +IER,KCYCLA,NDOUT,LKOUT,TMINT,NDAYS,NYEAR,KINTRA,MINTRA)
ISN      165            TMCYMX=0.D0
                  C...CHECK CONTROL PARAMETERS IN BOTH SCF AND PDF
ISN      166            IF(PDFFLG .NE. 0)CALL PDFHDR(KPDF,KDEBUG,NDAYS,NYEAR,MCYCLE,
                       +MINTRV,MCH,KEYPDF,PDFNAM,MPDFNM,TMINT,IER,NYPDF)
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:38      NAME: MAIN     PAGE:     7
                    *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN      168           DO 138 I=1,MINTRV
ISN      169    138    TMCYMX=TMCYMX+TMINT(I)
                C...CALCULATE THE NUMBER OF DAYS FROM 1900 TO THE STARTING DATE
ISN      170           CALL TMCALC(NYEAR,NDAYS,TMOFST)
ISN      171           GO TO 122
ISN      172    117    IF(JWRD .NE. HPHYSY)GO TO 118
                C...SUB-FILE PHYSYS
ISN      173           CALL PHSYS2(JWRD,IER,XSTR,XMET,JMET,
                      +USERNM,INTNBR,NDMAP,TYPCOD,MCH,KEY,KEYCHA,
                      +KEYCHD,KEYCHH,KEYCHI,KEYCHM,KEYCHN,KEYCHR,KEYCHC,KEYCD,KEYCN,
                      +KYRCHI,KYRCHM,DIVHD,VLMINR,VLINCR,ELVLR,KMTPHR,EWTPHR,AREAI,PRTNI,
                      +PRTNM,RHDPHH,QMAXH,QPPHH,ELFLHH,FLMNHH,FLICHH,ELFLTH,
                      +FLMNTH,FLICTH,KTPPHH,RNOFFA,FLMINA,VLMINA,KVLTMA,KTPPHC,KRFPHC,
                      +FLELC,ELMNC,ELICC,TRTNM,HDLPHH,ZMXFLD)
ISN      174           GO TO 122
ISN      175    118    IF(JWRD .NE. HPENSY)GO TO 119
                C...SUB-FILE PENSYS
ISN      176           CALL PNSYS2(JWRD,IER,MINTRV,INTNBR,NDMAP,TYPCOD,CHMAP,
                      +NODARC,ACOST,ARCLB,AFLOW,ARCUB,NSETS,MARCS,NCHA,NCHAB,NRTNA,
                      +KEYPCR,KEY,KEYPCI,KEYPCM,KEYPCH,KEYPCN,KEYPCA,KEYPCD,RCTMR,
                      +ZELTMR,ZFLPUM,ZPDPUH,RCTMN,ZFLTMN,ZFLPUA,RCTMD,ZFLTMD,MCH,
                      +ZMXFLD,ZMXVLD,RCTMI,ZFLTMI)
ISN      177           GO TO 122
ISN      178    119    IF(JWRD .NE. HWATDE)GO TO 120
                C...SUB-FILE WATDEM
ISN      179           CALL WTDM2(MINTRV,JWRD,IER,MARCS,AFLOW,NODARC,
                      +CHMAP,NDMAP,INTNBR,NSETS,NCHA,NCHAB,ARCLB,ARCUB,TYPCOD,MCH,KEY,
                      +KEYCHW,WDTMW,PDTMH,WDTMI,RFFTMI,DEFTMI,WDTMM,PRTNI,ZELPUI,AREAI,
                      +NRTNA,MPDFNM,PDFNAM,PDFFLG,KEYPDF,RFCTMI)
ISN      180           GO TO 122
ISN      181    120    IF(JWRD .NE. HWATSU)GO TO 121
                C...SUB-FILE WATSUP
ISN      182           CALL WTSP2(JWRD,MINTRV,IER,INTNBR,NDMAP,CHMAP,NCHA,
                      +NCHAB,AFLOW,ARCLB,ARCUB,ACOST,NSETS,MARCS,NODARC,TYPCOD,MCH,KEY,
                      +KEYCHF,WSTMF,KTPWSA,FFLTMA,EVPRTM,XTM,MECMHN,VLMINR,VLINCR,ELVLR,
                      +KEYPDF,PDFFLG,PDFNAM,MPDFNM,TMINT(1),KMTPDF,KINTRA,MINTRA,
                      +KTM,FTM,STOARC)
ISN      183           GO TO 122
                C...INCORRECT SUB-FILE SETUP
ISN      184    121    WRITE(KDEBUG,1003)
ISN      185           WRITE(KDEBUG,999)JWRD
ISN      186           STOP
ISN      187    122    IF(IER)124,115,123
ISN      188    123    WRITE(KDEBUG,1005)
ISN      189           WRITE(KDEBUG,999)JWRD
ISN      190           STOP
ISN      191    124    CONTINUE
                C...SET LIMITS ON ARC BETWEEN SBN AND SSN AND SET TYPCOD
```

```
LEVEL 1.4.1 (MAY 1985)            VS FORTRAN           DATE: MAY 02, 1988    TIME: 15:44:38       NAME: MAIN     PAGE:    8
                 *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN     192            TYPCOD(MCH)=11
ISN     193            ARCUB(NODARC(CHMAP(MCH,4)))=1.D04
ISN     194            ARCUB(NODARC(CHMAP(MCH,4))+1)=0.0
ISN     195            ACOST(NODARC(CHMAP(MCH,4)))=0.0
ISN     196            ACOST(NODARC(CHMAP(MCH,4))+1)=0.0
                 C...REPLACE USER NUMBERS WITH METKEY REFERENCES IN KMTPHR
ISN     197            CALL RPLMET(KMTPHR,MECMHN,KDEBUG)
                 C...BEGIN TIME LOOP
ISN     198            IPAGE=0
ISN     199            TMREL=0.D0
ISN     200            TMELA=0.D0
                 C...SET UP TIME PARAMETERS FOR APPORTIONMENT CHANNEL
ISN     201            IF (MCYCLA.NE.0.AND.MINTRA.NE.0)
                      +CALL SETTMA(KCYCLA,KINTRA,JCYCLA,JINTRA,MINTRV,MINTRA,MCYCLA,
                      +TMREL,TMINT,TMCYCA,TMINTA)
                 C...IF STARTING YEAR DIFFERS IN SCF AND PDF, SKIP THE EXTRA RECORDS
ISN     203            NYSKIP=NYEAR-NYPDF
ISN     204            IF(NYSKIP.EQ.0.OR.PDFFLG.EQ.0) GO TO 173
ISN     205            DO 171 NK=1,NYSKIP
ISN     206            DO 172 I=1,MINTRV
ISN     207            READ(KPDF,19) (XSCR4(I1),I1=1,MPDFNM)
ISN     208      172   CONTINUE
ISN     209      171   CONTINUE
ISN     210      19    FORMAT(1X,10F10.3)
                 C...DEBUG PRINT OF SUBROUTINE TOPOUT AND KEYOUT
ISN     211      173   IF(DUSWCH(1).EQ.0) GO TO 170
ISN     212            CALL TOPOUT(NDMAP,NSSNA,NSBNA,NRTNA,CHMAP,NCHA,NCHAB,SETMAP,
                      +INTNBR,MNDNBR,MCHNBR,MARCS,NSETS,MCH,NODARC,NEXNOD,NHEAD,NTAIL,
                      +NHIST,NLOWST,MND,TYPCOD)
ISN     213            CALL KEYOUT(KDEBUG,KEY,KEYCHA,KEYCHD,KEYCHH,KEYCHI,KEYCHM,KEYCHW,
                      +KEYCHN,KEYCHR,KEYCHF,KEYCHC,KEYCD,KEYCN,KYRCHI,KYRCHM,MCH)
ISN     214      170   MARCS2=MARCS*2
ISN     215            MND1=MND+1
                 C...REFORMULATE THE INTERNAL NETWORK TO CONFORM TO NEW OKA
ISN     216            CALL PSARCS(ACOST,MARCS,MND,MARCS2,MND1,Z,NSETS,NTAIL,
                      +NHEAD,PACOST,NCHA,SETMAP,MIDL,NODE,NLOWST,NHIST)
                 C...BEGIN CYCLE LOOP
ISN     217            DO 139 JCYCLE=1,MCYCLE
ISN     218            JCYCLA=1
ISN     219            JINTRA=1
ISN     220            JAPPT=0
ISN     221            TMCYC=0.D0
                 C...RE-SET ACCUMULATED VOLUME OF DIVERSION TO ZERO
ISN     222            DO 159 I=1,MNBPHD
ISN     223      159   TMXVLD(I)=0.D0
                 C...RETRIEVE TIME DEPENDENT VARIABLES FROM PDF
ISN     224            IF(PDFFLG .NE. 0)CALL PDFTM(KPDF,XSCR,MPDFNM,MINTRV,KEYPDF,MCH,
                      +TYPCOD,KEY,FFLTMA,PDTMH,WSTMF,WDTMI,WDTMM,WDTMW,EVPRTM,KMTPDF)
                 C...BEGIN CALCULATION TIME INTERVAL LOOP
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:38      NAME: MAIN     PAGE:     9
                        *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN       226              DO 140 JINTRV=1,MINTRV
ISN       227              ICOUNT=0
ISN       228              TMCYC=TMCYC+TMINT(JINTRV)
ISN       229              TMREL=TMREL+TMINT(JINTRV)
ISN       230              TMELA=TMELA+TMINT(JINTRV)
ISN       231              JREL=MINTRV*(JCYCLE-1)+JINTRV
ISN       232              TMV=TMCYC-TMINT(JINTRV)+1.
                     C...DEBUG PRINT OF CURRENT CYCLE AND INTERVAL
ISN       233              IF(DUSWCH(3).NE.0.OR.DUSWCH(4).NE.0)
                          +WRITE(KDEBUG,1007)JCYCLE,JINTRV
ISN       235        1007  FORMAT(1X,'MAINWB: BEGINNING CYCLE ',I4,5X,'INTERVAL ',I3,5X,
                          +25(1H#))
                     C...CHANNEL LOOP
ISN       236              DO 125 I=1,MCH
ISN       237              IX=IABS(TYPCOD(I))
ISN       238              GO TO(126,127,125,129,130,131,132,133,134,125,125),IX
                     C...APPORTIONMENT CHANNEL SECTION
ISN       239        126   CALL APPTTM(I,KEY,MCH,KCYCLA,MCYCLA,JINTRV,VLTGA,MINTRV,
                          +FFLTMA,RNOFFA,TMINT,MINTRA,JAPPT,KINTRA,VLACHA,FLMINA,NODARC,
                          +NSETS,CHMAP,ARCLB,ARCUB,MARCS,NCHA,TMREL,TMELA,TMINTA,TMCYCA,
                          +JCYCLA,ZFLPUA,JINTRA,KEYPCA)
ISN       240              NUMSUF=MINTRV*(KEY(I)-1)+JINTRV
ISN       241              J2SUF=NUMSUF/60+1
ISN       242              J2PSUF=NUMSUF/60
ISN       243              IF ((NUMSUF-60*J2PSUF).EQ.0) J2SUF=J2SUF-1
ISN       245              J1SUF=NUMSUF-60*(J2SUF-1)
ISN       246              TIDEAL(JINTRV,I)=FFLTMA(J1SUF,J2SUF)*RNOFFA(KEY(I))
ISN       247              GO TO 125
                     C...DIVERSION CHANNEL SECTION
ISN       248        127   CALL DIVTM(I,KEY,MCH,NCHA,TMCYC,ZFLTMD,RCTMD,CHMAP,NODARC,ARCUB,
                          +ARCLB,NSETS,MARCS,ZMXFLD,TMV,ZFLOWD,ZMXVLD,TMXVLD)
ISN       249              GO TO 125
                     C...NATURAL INFLOW SECTION
ISN       250        129   CALL INFLTM(I,JINTRV,MINTRV,INTNBR,NDMAP,CHMAP,NODARC,ARCLB,ARCUB,
                          +NSETS,MARCS,MCH,KEY,WSTMF)
ISN       251              GO TO 125
                     C...IRRIGATION CONSUMPTION CHANNEL SECTION
ISN       252        130   CALL IRRTM(I,JINTRV,MINTRV,INTNBR,NDMAP,CHMAP,NODARC,ARCLB,
                          +ARCUB,NSETS,MARCS,WDTMI,RFFTMI,DEFTMI,MCH,KEY,PRTNI,ZELPUI,
                          +AREAI,NCHA,NRTNA,KDEBUG,KEYPCI,TMINT(JINTRV),KWRITE,RFCTMI,
                          +TMCYC,TMV,RCTMI,ZFLTMI)
ISN       253              GO TO 125
                     C...MAJOR WITHDRAWAL SECTION
ISN       254        131   CALL MAJTM(I,JFLGM,JINTRV,MINTRV,INTNBR,NDMAP,CHMAP,NODARC,
                          +ARCLB,ARCUB,NSETS,MARCS,MCH,NCHA,NRTNA,KEY,WDTMM,ZFLPUM,TRTNM,
                          +PRTNM,KDEBUG,KEYPCM,KWRITE)
ISN       255              GO TO 125
                     C...MINOR WITHDRAWAL SECTION
```

```
                  *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN       256   132   CALL WITHTM(I,JINTRV,WDTMW,INTNBR,NDMAP,CHMAP,NODARC,ARCUB,ARCLB,
                     +MINTRV,NSETS,MARCS,KEY,MCH)
ISN       257         GO TO 125
                C...NATURAL CHANNEL SECTION
ISN       258   133   CALL NATTM(I,KEY,NCHA,MCH,TMCYC,ZFLOWN,KDEBUG,RCTMN,ZFLTMN,
                     +CHMAP,NODARC,ARCLB,ARCUB,NSETS,MARCS,NCHAB,TMV)
ISN       259         GO TO 125
                C...RESERVOIR SECTION
ISN       260   134   CALL RESTM(I,VLMINR,VLINCR,KMTPHR,EWTPHR,EVPRTM,NODARC,CHMAP,
                     +ARCUB,ARCLB,NCHA,NCHAB,TMINT(JINTRV),JINTRV,ELVLR,KDEBUG,NSETS,
                     +MARCS,MCH,KEY,MINTRV,GNMET,RCTMR,ZELEVS,ZELTMR,TMCVMX,TMCVC)
ISN       261   125   CONTINUE
ISN       262         JAPPT=1
ISN       263         DO 128 I=1,MCH
ISN       264         IF(TYPCOD(I) .EQ. 3)GO TO 136
ISN       265         IF(TYPCOD(I) .LT. 0)GO TO 137
ISN       266         GO TO 128
                C...HYDROPOWER CHANNEL SECTION
ISN       267   136   CALL HYDRTM(I,KEY,MCH,QMAXH,QPPHH,PDTMH,RHDPHH,ELFLHH,FLMNHH,
                     +FLICHH,MINTRV,ELFLTH,FLMNTH,FLICTH,ARCUB,MARCS,TMINT(JINTRV),
                     +VLMINR,VLINCR,ELVLR,HDLPHH,NODARC,KEYPCH,ARCLB,NSETS,CHMAP,NCHA,
                     +ZPDPUH,INTNBR,NCHAB,NDMAP,KTPPHH,JINTRV,KDEBUG,HDROP,
                     +JREL,OUTFIL,2,FHOLD,FHASM,FXOLD,HDOLD,KWRITE)
ISN       268         GO TO 128
                C...CONTROL STRUCTURE SECTION
ISN       269   137   CALL CTLTM(I,KEYCHC,KTPPHC,CHMAP,NDMAP,INTNBR,KRFPHC,KEY,NODARC,
                     +NCHAB,ARCUB,TMINT(JINTRV),VLMINR,VLINCR,ELVLR,FLELC,TYPCOD,ZMXFLD,
                     +ARCLB,NCHA,KDEBUG,NSETS,MCH,MARCS,ELMNC,ELICC,JINTRV,FWOLD,2,
                     +OUTDAT,MINTRV,CTLHD,CTLFX)
ISN       270   128   CONTINUE
                C...BEGIN KILTER PROCESSING
ISN       271   151   CALL ARRAYS(MND,MND1,MARCS,MARCS2,MIDL,NODE,NDARCS,
                     +NTAIL,ARCUB,AFLOW,ARCLB,JNODE,PANC,R,PACOST)
ISN       272         CALL KILTER(MND,MND1,MARCS,MARCS2,MIDL,NODE,NDARCS,LABEL,INFES,
                     +NTAIL,NHEAD,ARCUB,AFLOW,JNODE,PANC,R,PACOST,INLOC,VNODE,*141)
                C...DEBUG PRINT OF SUBROUTINE ARCCHK
ISN       273   556   IF (DUSWCH(2).EQ.0) GO TO 155
ISN       274         IF(JREL.GE.LKH1.AND.JREL.LE.LKH2)
                     +CALL ARCCHK(0,NVEAR+JCYCLE-1,JINTRV,KDEBUG,NSETS,NODARC,SETMAP,
                     +CHMAP,NCHA,NDMAP,ARCUB,AFLOW,ARCLB,ACOST,TYPCOD,MARCS,PACOST,
                     +NCHAB,MARCS2,R)
ISN       276   155   CONTINUE
                C...ORGANIZE THE RAW OUTPUT FROM KILTER
ISN       277         CALL PASSTR(AFLOW,NCHA,NCHAB,CHMAP,NODARC,TYPCOD,ZN,OUTDAT,
                     +NSETS,MCH,MINTRV,MARCS,JINTRV,VLMINR,VLINCR,ELVLR,KEY,OUTFIL,
                     +DEFTMI,AREAI,TMINT,GNMET,TMREL,TMOFST,NDYOUT,HDROP,QPPHH,HDLPHH,
                     +NMOOUT,NYROUT,ARCLB,ARCUB,VLACHA,KCYCLA,MCYCLA,JCYCLA,APPVOL,
                     +JCYCLE,OUTEVP)
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8
                C...CHECK HEAD ON CONTROL STRUCTURE AND HYDRO PLANT, RE-ITERATE IF NECESSA
ISN       278           IER=0
ISN       279           DO 143 I=1,MCH
ISN       280           IF(TYPCOD(I).GE.0) GO TO 144
ISN       281           IF(TYPCOD(I).EQ.-2) GO TO 59
ISN       282           CALL NATTM(I,KEY,NCHA,MCH,TMCYC,ZFLOWN,KDEBUG,RCTMN,ZFLTMN,
                       +CHMAP,NODARC,ARCLB,ARCUB,NSETS,MARCS,NCHAB,TMV)
ISN       283           GO TO 60
ISN       284     59    CALL DIVTM(I,KEY,MCH,NCHA,TMCYC,ZFLTMD,RCTMD,CHMAP,NODARC,ARCUB,
                       +ARCLB,NSETS,MARCS,ZMXFLD,TMV,ZFLOWD,ZMXVLD,TMXVLD)
ISN       285     60    CALL CTLTM(I,KEYCHC,KTPPHC,CHMAP,NDMAP,INTNBR,KRFPHC,KEY,NODARC,
                       +NCHAB,ARCUB,TMINT(JINTRV),VLMINR,VLINCR,ELVLR,FLELC,TYPCOD,ZMXFLD,
                       +ARCLB,NCHA,KDEBUG,NSETS,MCH,MARCS,ELMNC,ELICC,JINTRV,FWOLD,IER,
                       +OUTDAT,MINTRV,CTLMD,CTLFX)
ISN       286     144   IF(TYPCOD(I).NE.3) GO TO 143
ISN       287           CALL HYDRTM(I,KEY,MCH,QMAXH,QPPHH,PDTMH,RHDPHH,ELFLHH,FLMNHH,
                       +FLICHH,MINTRV,ELFLTH,FLMNTH,FLICTH,ARCUB,MARCS,TMINT(JINTRV),
                       +VLMINR,VLINCR,ELVLR,HDLPHH,NODARC,KEYPCH,ARCLB,NSETS,CHMAP,NCHA,
                       +ZPDPUH,INTNBR,NCHAB,NDMAP,KTPPHH,JINTRV,KDEBUG,HDROP,
                       +JREL,OUTFIL,IER,FHOLD,FHASM,FXOLD,HDOLD,KWRITE)
ISN       288     143   CONTINUE
                C...ICOUNT IS THE NUMBER OF ITERATION IN ONE TIME INTERVAL
ISN       289           ICOUNT=ICOUNT+1
ISN       290           IF (ICOUNT.LT.6 ) GO TO 150
ISN       291           WRITE (KPRINT,1008)
ISN       292     1008  FORMAT(1H ,//,1H ,' MAINWB : WARNING - MORE THAN 5 ITERATIONS')
ISN       293     150   IF (IER.EQ.0.OR.ICOUNT.GE.6) GO TO 157
                C...RE-ITERATION REQUIRED, RESET THE RESERVOIR STORAGE TO THE STARTING STO
ISN       294           DO 156 I=1,MCH
ISN       295           IF(TYPCOD(I).NE.9) GO TO 156
ISN       296           JARC=NODARC(CHMAP(I,4))+NCHAB(I)+1
ISN       297           ARCUB(JARC)=STOARC(I)
ISN       298           ARCLB(JARC)=STOARC(I)
ISN       299     156   CONTINUE
ISN       300            GO TO 151
                C...NO RE-ITERATION REQUIRED, UPDATE THE CURRENT CONDITION
ISN       301     157   DO 154 I=1,MCH
ISN       302           IF(TYPCOD(I).NE.9) GO TO 158
                C...UPDATE THE RESERVOIR STORAGE ARC
ISN       303           IX=JINTRV+1
ISN       304           IF(JINTRV .EQ. MINTRV)IX=1
ISN       306           JARC=NODARC(CHMAP(I,4))+NCHAB(I)+1
ISN       307           STOARC(I)=ARCUB(JARC)*TMINT(JINTRV)/TMINT(IX)
ISN       308           ARCUB(JARC)=STOARC(I)
ISN       309           ARCLB(JARC)=STOARC(I)
ISN       310           GO TO 154
ISN       311     158   IF(TYPCOD(I).NE.1) GO TO 160
                C...UPDATE THE TOTAL VOLUME OF APPORTIONMENT FLOW
```

```
LEVEL 1.4.1 (MAY 1985)              VS FORTRAN             DATE: MAY 02, 1988     TIME: 15:44:43                      PAGE:    22
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK    SOURCE    TERM    OBJECT FIXED  NOTEST NOTRMFLG    SRCFLG
ISN       312              VLACHA(KEY(I))=VLACHA(KEY(I))+APPVOL(KEY(I))*
                          +TMINT(JINTRV)*8.64D4
ISN       313              IF(JINTRV.EQ.KCYCLA(JCYCLA)) VLACHA(KEY(I))=0.D0
ISN       315              GO TO 154
ISN       316       160    IF(TYPCOD(I).NE.2) GO TO 154
                    C...UPDATE THE TOTAL VOLUME OF DIVERSION
ISN       317              CALL CHNFLW(OUTDAT,JINTRV,MINTRV,MCH,I,FWDJ)
ISN       318              IVZX=KEY(I)
ISN       319              TMXVLD(IVZX)=TMXVLD(IVZX)+FWDJ*TMINT(JINTRV)*86.4
ISN       320       154    CONTINUE
ISN       321              IF(JINTRV .LT. KCYCLA(JCYCLA))GO TO 142
ISN       322              JCYCLA=JCYCLA+1
ISN       323              TMELA=0.D0
ISN       324              JAPPT=0
ISN       325       142    IF(JINTRV .EQ. KINTRA(JINTRA))JINTRA=JINTRA+1
                    C...SAVE THE LOWER BOUND OF THE IDEAL FLOW ARC AS THE IDEAL VALUE
ISN       327              DO 200 I=1,MCH
ISN       328              IF (TYPCOD(I).EQ.1) GO TO 200
ISN       329              TIDEAL(JINTRV,I)=ARCLB(NODARC(CHMAP(I,4))+NCHAB(I))
ISN       330       200    CONTINUE
ISN       331       140    CONTINUE
                    C...NORMAL PRINT OF OUTPUT FROM KILTER
ISN       332              CALL PGOUT(TITLE,DATE,REFNAM,USER,OUTFIL,OUTDAT,ZN,USERNM,
                          +MINTRV,MOUTND,MOUTLK,MCH,NDOUT,LKOUT,CHMAP,TYPCOD,NDVOUT,NMOOUT,
                          +NVROUT,IPAGE,JCYCLE,MOUTAL,TIDEAL,OUTEVP,MOUTEV,MNBPHR,KEYCHR,
                          +TMINT)
ISN       333       139    CONTINUE
ISN       334              STOP
                    C...ABNORMAL TERMINATION SECTION
ISN       335       141    IF(DUSWCH(5).EQ.0) GO TO 601
ISN       336              I1=1
ISN       337              CALL PGOUT(TITLE,DATE,REFNAM,USER,OUTFIL,OUTDAT,ZN,USERNM,
                          +MINTRV,MOUTND,MOUTLK,MCH,NDOUT,LKOUT,CHMAP,TYPCOD,NDVOUT,NMOOUT,
                          +NVROUT,IPAGE,JCYCLE,I1,TIDEAL,OUTEVP,MOUTEV,MNBPHR,KEYCHR,
                          +TMINT)
ISN       338              CALL TOPOUT(NDMAP,NSSNA,NSBNA,NRTNA,CHMAP,NCHA,NCHAB,SETMAP,
                          +INTNBR,MNDNBR,MCHNBR,MARCS,NSETS,MCH,NODARC,NEXNOD,NHEAD,NTAIL,
                          +NHIST,NLOWST,MND,TYPCOD)
ISN       339              CALL ARCCHK(1,NYEAR+JCYCLE-1,JINTRV,KDEBUG,NSETS,NODARC,SETMAP,
                          +CHMAP,NCHA,NDMAP,ARCUB,AFLOW,ARCLB,ACOST,TYPCOD,MARCS,PACOST,
                          +NCHAB,MARCS2,R)
ISN       340              GO TO 602
ISN       341       601    CALL ARCCHK(2,NYEAR+JCYCLE-1,JINTRV,KDEBUG,NSETS,NODARC,SETMAP,
                          +CHMAP,NCHA,NDMAP,ARCUB,AFLOW,ARCLB,ACOST,TYPCOD,MARCS,PACOST,
                          +NCHAB,MARCS2,R)
ISN       342       602    STOP
ISN       343              END
*STATISTICS*    SOURCE STATEMENTS = 328, PROGRAM SIZE = 1739920 BYTES, PROGRAM NAME = MAIN   .     PAGE:     1.
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988     TIME: 15:44:48                         PAGE:    54
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE   TERM   OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
*STATISTICS*    NO DIAGNOSTICS GENERATED.
**MAIN** END OF COMPILATION 1 ******
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:48                    PAGE:    54
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE   TERM   OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
                    NOSYM NORENT   SDUMP AUTODBL(NONE)   NOSXM   IL
                     OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)      CHARLEN(500)
                 *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN        1           SUBROUTINE KILTER(MND,MND1,MARCS,MARCS2,MIDL,NODE,NDARCS,
                      +LABEL,INFES,NTAIL,NHEAD,ARCUB,AFLOW,JNODE,PANC,R,
                      +PACOST,INLOC,VNODE,*)
                 C...SUBROUTINE GENERATES AN OPTIMUM CIRCULATION IN A NETWORK
                 C...PREVIOUSLY REDEFINED BY THE SUBROUTINE  PSARCS.
ISN        2           INTEGER*4 NTAIL(MARCS2),NHEAD(MARCS2),NODE(MND1),
                      +MIDL(MND),JNODE(MARCS2),NDARCS(MARCS2),PACOST(MARCS2),
                      +INLOC(MND),PACCHG,PACOS1,LABEL(MND)
ISN        3           REAL*8 ARCUB(MARCS),AFLOW(MARCS),R
ISN        4           REAL*8 PANC(MARCS2),FIP,FLWCHG,PANC1,VNODE(MND)
                 C...ENTER THE MAIN LOOP. EXAMINE THE KILTER STATUS OF EACH ARC
                 C...AND ITS MIRROR AND DETERMINE THE DESIRED CHANGE.
ISN        5           DO 20 M=1,MARCS
ISN        6           N=M+MARCS
ISN        7           DO 20 KARC=1,2
ISN        8           IF(KARC.EQ.2)THEN
ISN        9           IM=N
ISN       10           IMIR=M
ISN       11           ELSE
ISN       12           IM=M
ISN       13           IMIR=N
ISN       14           ENDIF
ISN       15     22    IF(PANC(IM).LT.-R)THEN
ISN       16           IT=NTAIL(IM)
ISN       17           IS=NHEAD(IM)
ISN       18           KARCS=IMIR
ISN       19           ELSE IF(PANC(IM).GT.R.AND.PACOST(IM).LT.0)THEN
ISN       20           IT=NHEAD(IM)
ISN       21           IS=NTAIL(IM)
ISN       22           KARCS=IM
ISN       23           ELSE IF(PANC(IMIR).LT.-R)THEN
ISN       24           IT=NHEAD(IM)
ISN       25           IS=NTAIL(IM)
ISN       26           KARCS=IM
ISN       27           ELSE
ISN       28           GO TO 20
ISN       29           ENDIF
                 C...ENTER LABELING SECTION. INITIALIZE PARAMETERS.
ISN       30           NLN=1
ISN       31           NS=0
ISN       32           NLNCRS=1
ISN       33           LABEL(IT)=KARCS
ISN       34           INLOC(NLN)=IT
ISN       35           NNLS1=1
                 C...ENTER LABELING LOOP. EXIT LOOP TO EITHER BREAKTHROUGH OR
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:48      NAME: KILTER  PAGE:   55
                   *....*...1.........2.........3.........4.........5.........6.........7.*.......8
                   C...POTENTIAL CHANGE SECTION.
ISN        36      30    CALL LBLRTN(IT;IS,NODE,MIDL,NDARCS,NHEAD,MARCS2,
                        +MND,MND1,NNLS1,LABEL,INLOC,NLN,NNLS,*73)
                   C...BREAKTHROUGH SECTION ...
                   C...DETERMINE THE AMOUNT OF PERMISSIBLE FLOW CHANGE.
ISN        37      51    FLWCHG=1.D08
ISN        38            NSN=IS
ISN        39            DO 59 JAC=1,MND
ISN        40            NPA=LABEL(NSN)
ISN        41            IF(NPA.GT.MARCS)THEN
ISN        42            NMA=NPA-MARCS
ISN        43            ELSE
ISN        44            NMA=NPA+MARCS
ISN        45            ENDIF
ISN        46            IF(PACOST(NPA).LE.0)THEN
ISN        47            FIP=PANC(NPA)
ISN        48            VNODE(JAC)=FIP
ISN        49            ELSE
ISN        50            FIP=-PANC(NMA)
ISN        51            VNODE(JAC)=FIP
ISN        52            ENDIF
ISN        53            FLWCHG=DMIN1(FLWCHG,FIP)
ISN        54            NSN=NTAIL(NPA)
ISN        55            IF(NSN.EQ.IS)GO TO 55
ISN        56      59    CONTINUE
                   C...CHANGE THE FLOW ALONG THE FLOW AUGMENTING PATH.
ISN        57      55    DO 61 JAC=1,MND
ISN        58            NPA=LABEL(NSN)
ISN        59            IF(NPA.GT.MARCS)THEN
ISN        60            NMA=NPA-MARCS
ISN        61            ELSE
ISN        62            NMA=NPA+MARCS
ISN        63            ENDIF
ISN        64            PANC1=PANC(NMA)
ISN        65            PANC(NPA)=PANC(NPA)-FLWCHG
ISN        66            PANC(NMA)=PANC(NMA)+FLWCHG
ISN        67            IF(PACOST(NMA).NE.0)GO TO 65
ISN        68            IF(PANC1.GT.R.OR.PANC(NMA).LE.R)GO TO 65
ISN        69            CALL QINP(NSN,NMA,MND,MND1,MARCS2,MIDL,NODE,NDARCS)
ISN        70      65    IF(VNODE(JAC).EQ.FLWCHG)CALL PINQ(NHEAD(NMA),NPA,MND,MND1,MARCS2,
                        +MIDL,NODE,NDARCS)
ISN        72            NSN=NTAIL(NPA)
ISN        73            IF(NSN.EQ.IS)THEN
                   C...DELETE LABELS AND CHECK THE KILTER STATUS OF AN ARC.
ISN        74            DO 71 I=1,NLN
ISN        75      71    LABEL(INLOC(I))=0
ISN        76            GO TO 22
ISN        77            ELSE
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:48      NAME: KILTER  PAGE:   56
                  *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN        78            ENDIF
ISN        79      61    CONTINUE
                  C...POTENTIAL CHANGE SECTION.
ISN        80      73    NSLPCH=NS
ISN        81            NRLPCH=NR
ISN        82            PACCHG=1E09
ISN        83            NBRK=0
ISN        84            NS=0
ISN        85            NR=MARCS2
ISN        86            IF(NSLPCH.EQ.0)GO TO 75
ISN        87            IF(NRLPCH.EQ.MARCS2)GO TO 74
                  C...NR SET RECYCLING SECTION.
ISN        88            DO 76 J=MARCS2,NRLPCH+1,-1
ISN        89            JL=JNODE(J)
ISN        90            JNEXT=NHEAD(JL)
ISN        91            IF(LABEL(JNEXT).NE.0)GO TO 76
ISN        92            JNODE(NR)=JL
ISN        93            NR=NR-1
ISN        94      76    CONTINUE
                  C...NS SET RECYCLING SECTION.
ISN        95      74    DO 77 J=1,NSLPCH
ISN        96            JL=JNODE(J)
ISN        97            JNEXT=NHEAD(JL)
ISN        98            IF(LABEL(JNEXT).NE.0)GO TO 77
ISN        99            IF(PACOST(JL).GT.0)THEN
ISN       100            NS=NS+1
ISN       101            JNODE(NS)=JL
ISN       102            PACCHG=MIN(PACCHG,PACOST(JL))
ISN       103            ELSE
ISN       104            JNODE(NR)=JL
ISN       105            NR=NR-1
ISN       106            ENDIF
ISN       107      77    CONTINUE
ISN       108      75    IF(NLNCRS.GT.NLN)GO TO 90
                  C...DETERMINE THE MIN MARGINAL COST CHANGE.
ISN       109            DO 80 ND=NLNCRS,NLN
ISN       110            I=INLOC(ND)
ISN       111            NSTART=MIDL(I)+1
ISN       112            NEND=NODE(I+1)-1
ISN       113            IF(NSTART.GT.NEND)GO TO 80
ISN       114            DO 81 JP=NSTART,NEND
ISN       115            NA=NDARCS(JP)
ISN       116            IARC=NHEAD(NA)
ISN       117            IF(LABEL(IARC).NE.0)GO TO 81
ISN       118            IF(PANC(NA).LT.-R.OR.PACOST(NA).LE.0)THEN
ISN       119            JNODE(NR)=NA
ISN       120            NR=NR-1
ISN       121            ELSE
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:48     NAME: KILTER  PAGE:   57
                        *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN      122            NS=NS+1
ISN      123            JNODE(NS)=NA
ISN      124            PACCHG=MIN(PACCHG,PACOST(NA))
ISN      125            ENDIF
ISN      126     81     CONTINUE
ISN      127     80     CONTINUE
ISN      128     90     NLNCRS=NLN+1
                 C...DETERMINE IF PROBLEM IS  INFEASIBLE.
ISN      129            IF(NS.EQ.0)THEN
ISN      130            INFES=1
ISN      131            GO TO 33
ISN      132            ELSE
ISN      133            ENDIF
                 C...CHANGE OF MARGINAL COST FOR SET S.
ISN      134            DO 91 I=1,NS
ISN      135            NPA=JNODE(I)
ISN      136            IF(NPA.GT.MARCS)THEN
ISN      137            NMA=NPA-MARCS
ISN      138            ELSE
ISN      139            NMA=NPA+MARCS
ISN      140            ENDIF
ISN      141            PACOST(NPA)=PACOST(NPA)-PACCHG
ISN      142            PACOST(NMA)=-PACOST(NPA)
ISN      143            IF(PACOST(NPA).NE.0.OR.PANC(NPA).LE.R)GO TO 91
ISN      144            CALL QINP(NTAIL(NPA),NPA,MND,MND1,MARCS2,MIDL,NODE,NDARCS)
ISN      145            NLBA=NHEAD(NPA)
ISN      146            IF(LABEL(NLBA).NE.0)GO TO 91
ISN      147            LABEL(NLBA)=NPA
ISN      148            NLN=NLN+1
ISN      149            INLOC(NLN)=NLBA
ISN      150            IF(NLBA.EQ.IS)NBRK=1
ISN      152     91     CONTINUE
                 C...CHANGE OF MARGINAL COST FOR SET R.
ISN      153            IF(NR.EQ.MARCS2)GO TO 99
ISN      154            DO 92 I=NR+1,MARCS2
ISN      155            NPA=JNODE(I)
ISN      156            IF(NPA.GT.MARCS)THEN
ISN      157            NMA=NPA-MARCS
ISN      158            ELSE
ISN      159            NMA=NPA+MARCS
ISN      160            ENDIF
ISN      161            PACOS1=PACOST(NPA)
ISN      162            PACOST(NPA)=PACOS1-PACCHG
ISN      163            PACOST(NMA)=-PACOST(NPA)
ISN      164            IF(PACOS1.LT.0.OR.PACOST(NPA).GE.0)GO TO 92
ISN      165            IF(PANC(NPA).LT.-R.OR.PANC(NPA).GT.R)GO TO 92
ISN      166            CALL PINQ(NHEAD(NPA),NMA,MND,MND1,MARCS2,MIDL,NODE,NDARCS)
ISN      167     92     CONTINUE
```

```
LEVEL 1.4.1 (MAY 1985)             VS FORTRAN           DATE: MAY 02, 1988    TIME: 15:44:48       NAME: KILTER  PAGE:   58
*STATISTICS*    NO DIAGNOSTICS GENERATED.
                C...OUT OF KILTER CHECK.
ISN       168   99     IF(PANC(IM).LT.-R.OR.PANC(IMIR).LT.-R)THEN
                C...CHECK BREAKTHROUGH AS A RESULT OF POTENTIAL CHANGE.
ISN       169          IF(NBRK.EQ.1)GO TO 51
ISN       170          NNLS1=NNLS
ISN       171          IF(NNLS1.EQ.NLN)GO TO 73
ISN       172          NNLS1=NNLS1+1
ISN       173          GO TO 30
ISN       174          ELSE IF(PANC(IM).GT.R.AND.PACOST(IM).LT.0)THEN
ISN       175          IF(NBRK.EQ.1)GO TO 51
ISN       176          NNLS1=NNLS
ISN       177          IF(NNLS1.EQ.NLN)GO TO 73
ISN       178          NNLS1=NNLS1+1
ISN       179          GO TO 30
ISN       180          ELSE
                C...ERASE LABELS TO PREPARE FOR THE NEXT ARC.
ISN       181          DO 98 I=1,NLN
ISN       182   98     LABEL(INLOC(I))=0
ISN       183          ENDIF
ISN       184   20     CONTINUE
                C...CONVERT FLOWS BACK INTO ACTUAL VALUES.
ISN       185   33     DO 32 M=1,MARCS
ISN       186          AFLOW(M)=ARCUB(M)-PANC(M)
ISN       187   32     CONTINUE
ISN       188          IF(INFES.EQ.1)RETURN 1
ISN       190          RETURN
ISN       191          END
*STATISTICS*    SOURCE STATEMENTS = 188, PROGRAM SIZE = 7816 BYTES, PROGRAM NAME = KILTER.    PAGE:   54.
*STATISTICS*     NO DIAGNOSTICS GENERATED.
**KILTER** END OF COMPILATION 26 ******
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:43                          PAGE:   22
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK    SOURCE    TERM    OBJECT FIXED  NOTEST NOTRMFLG    SRCFLG
                    NOSVM NORENT   SDUMP AUTODBL(NONE)   NOSXM    IL
                      OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)       CHARLEN(500)
                 *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN        1           SUBROUTINE ARRAYS(MND,MND1,MARCS,MARCS2,MIDL,NODE,NDARCS,
                      +NTAIL,ARCUB,AFLOW,ARCLB,JNODE,PANC,R,PACOST)
                 C...SUBROUTINE PERFORMS SETUP OF ARRAYS NDARCS AND MIDL.
ISN        2           INTEGER*4 NTAIL(MARCS2),NODE(MND1),
                      +MIDL(MND),JNODE(MARCS2),NDARCS(MARCS2),PACOST(MARCS2)
ISN        3           REAL*8 ARCUB(MARCS),AFLOW(MARCS),ARCLB(MARCS),R,PANC(MARCS2)
                 C...SUBROUTINE PERFORMS SETUP OF ARRAYS MIDL AND NDARCS.
ISN        4           DO 1 M=1,MARCS
ISN        5           N=M+MARCS
ISN        6           PANC(M)=ARCUB(M)-AFLOW(M)
ISN        7           PANC(N)=AFLOW(M)-ARCLB(M)
ISN        8     1     CONTINUE
ISN        9           DO 2 K=1,MND
ISN       10           JNODE(K)=NODE(K)
ISN       11     2     MIDL(K)=NODE(K+1)-1
ISN       12           JNODE(MND1)=MARCS2+1
                 C...THIS SECTION CREATES ARRAYS NDARCS AND MIDL. NDARCS IS AN ARRAY
                 C...OF OUTGOING PSEUDO-ARCS FOR EACH NODE, WHILE MIDL AND NODE ARE
                 C...VECTORS THAT SPLIT NDARCS SUB-SETS (ASSOCIATED WITH EACH
                 C...NODE) INTO SETS OF LABEL ELIGIBLE AND INELIGIBLE ARCS.
ISN       13           DO 3 M=1,MARCS2
ISN       14           I=NTAIL(M)
ISN       15           IF(M.GT.MARCS)THEN
ISN       16           MB=M-MARCS
ISN       17           ELSE
ISN       18           MB=M+MARCS
ISN       19           ENDIF
ISN       20           IF(PACOST(M).GT.0.AND.PANC(MB).LT.-R)THEN
ISN       21           IC=JNODE(I)
ISN       22           JNODE(I)=IC+1
ISN       23           ELSE IF(PACOST(M).LE.0.AND.PANC(M).GT.R)THEN
ISN       24           IC=JNODE(I)
ISN       25           JNODE(I)=IC+1
ISN       26           ELSE
ISN       27           IC=MIDL(I)
ISN       28           MIDL(I)=IC-1
ISN       29           ENDIF
ISN       30           NDARCS(IC)=M
ISN       31     3     CONTINUE
ISN       32           RETURN
ISN       33           END
*STATISTICS*   SOURCE STATEMENTS = 33, PROGRAM SIZE = 2668 BYTES, PROGRAM NAME = ARRAYS.    PAGE:   22.
*STATISTICS*    NO DIAGNOSTICS GENERATED.
**ARRAYS** END OF COMPILATION 7 ******
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:48                 PAGE:    59
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE    TERM    OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
                    NOSYM NORENT   SDUMP AUTODBL(NONE)   NOSXM    IL
                     OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)       CHARLEN(500)
               *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN        1          SUBROUTINE LBLRTN(IT,IS,NODE,MIDL,NDARCS,NHEAD,MARCS2,
                     +MND,MND1,NNLS1,LABEL,INLOC,NLN,NNLS,*)
               C...SUBROUTINE ATTEMPTS TO LABEL A FLOW AUGMENTING PATH.
ISN        2          INTEGER*4 LABEL(MND),INLOC(MND),MIDL(MND),NODE(MND1),
                     +NDARCS(MARCS2),NHEAD(MARCS2)
ISN        3          DO 1 NNLS=NNLS1,MND
ISN        4          I=INLOC(NNLS)
ISN        5          NSTART=NODE(I)
ISN        6          NEND=MIDL(I)
ISN        7          IF(NEND.LT.NSTART)GO TO 3
ISN        8          DO 2 JP=NSTART,NEND
ISN        9          JARC=NDARCS(JP)
ISN       10          NEWNOD=NHEAD(JARC)
ISN       11          IF(LABEL(NEWNOD).NE.0)GO TO 2
ISN       12          LABEL(NEWNOD)=JARC
ISN       13          NLN=NLN+1
ISN       14          INLOC(NLN)=NEWNOD
               C...REGULAR RETURN INDICATES A BREAKTHROUGH.
ISN       15          IF(NEWNOD.EQ.IS)RETURN
ISN       17   2      CONTINUE
ISN       18   3      IF(NNLS.EQ.NLN)GO TO 5
ISN       19   1      CONTINUE
               C...LABELED RETURN INDICATES A POTENTIAL CHANGE.
ISN       20   5      RETURN 1
ISN       21          END
*STATISTICS*   SOURCE STATEMENTS = 20, PROGRAM SIZE = 1764 BYTES, PROGRAM NAME = LBLRTN.     PAGE:    59.
*STATISTICS*    NO DIAGNOSTICS GENERATED.
**LBLRTN** END OF COMPILATION 27 ******
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:44:58                PAGE:   94
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE   TERM   OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
                    NOSYM NORENT   SDUMP AUTODBL(NONE)   NOSXM   IL
                     OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)      CHARLEN(500)
                 *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN        1           SUBROUTINE PINQ(NSN1,NPA,MND,MND1,MARCS2,MIDL,NODE,NDARCS)
                 C...SUBROUTINE MOVES AN ARC FROM SUBSET P INTO SUBSET Q.
ISN        2           INTEGER*4 MIDL(MND),NODE(MND1),NDARCS(MARCS2)
ISN        3           MSTART=NODE(NSN1)
ISN        4           MEND=MIDL(NSN1)
ISN        5           DO 1 IAC=MSTART,MEND
ISN        6           IF(NDARCS(IAC).EQ.NPA)THEN
ISN        7           ND1=NDARCS(MEND)
ISN        8           NDARCS(MEND)=NPA
ISN        9           NDARCS(IAC)=ND1
ISN       10           MIDL(NSN1)=MEND-1
ISN       11           RETURN
ISN       12           ELSE
ISN       13           ENDIF
ISN       14     1     CONTINUE
ISN       15           RETURN
ISN       16           END
*STATISTICS*   SOURCE STATEMENTS = 16, PROGRAM SIZE = 1260 BYTES, PROGRAM NAME = PINQ  .    PAGE:   94.
*STATISTICS*    NO DIAGNOSTICS GENERATED.
**PINQ** END OF COMPILATION 41 ******
```

```
LEVEL 1.4.1 (MAY 1985)            VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:45:07                PAGE:   108
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE   TERM   OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
                    NOSYM NORENT   SDUMP AUTODBL(NONE)   NOSXM   IL
                      OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)       CHARLEN(500)
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN        1          SUBROUTINE PSARCS(ACOST,MARCS,MND,MARCS2,MND1,Z,NSETS,NTAIL,
                     +NHEAD,PACOST,NCHA,SETMAP,MIDL,NODE,NLOWST,NHIST)
                C...SUBROUTINE TRANSFORMS THE ORIGINAL CAPACITATED NETWORK INTO
                C..." PSEUDO-ARCS " NETWORK AS REQUIRED BY THE REFORMULATED OUT-
                C...OF-KILTER ROUTINE.
ISN        2          INTEGER*4 NTAIL(MARCS2),NHEAD(MARCS2),NODE(MND1),SETMAP(NSETS),
                     +MIDL(MND1),PACOST(MARCS2),NCHA(321),NLOWST(MND),NHIST(MND)
ISN        3          REAL*8 ACOST(MARCS),Z
                C...THIS SECTION CREATES PSEUDO ARCS.
ISN        4          DO 1 M=1,MARCS
ISN        5          N=M+MARCS
ISN        6          PACOST(M)=IDNINT(ACOST(M)*Z)
ISN        7          PACOST(N)=-PACOST(M)
ISN        8          I=NTAIL(M)
ISN        9          J=NHEAD(M)
ISN       10          NHEAD(N)=I
ISN       11          NTAIL(N)=J
ISN       12    1     CONTINUE
ISN       13          NSUM=1
ISN       14          NODE(1)=NSUM
ISN       15          DO 2 I=1,MND
ISN       16          NSTRT=NLOWST(I)
ISN       17          NEND=NHIST(I)
ISN       18          DO 3 L=NSTRT,NEND
ISN       19          NSUM=NSUM+NCHA(SETMAP(L))
ISN       20    3     CONTINUE
ISN       21          NODE(I+1)=NSUM
ISN       22    2     CONTINUE
ISN       23          RETURN
ISN       24          END
*STATISTICS*    SOURCE STATEMENTS = 24, PROGRAM SIZE = 2216 BYTES, PROGRAM NAME = PSARCS.    PAGE:  108.
*STATISTICS*    NO DIAGNOSTICS GENERATED.
**PSARCS** END OF COMPILATION 44 ******
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:45:07                    PAGE:   109
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE   TERM   OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
                    NOSYM NORENT   SDUMP AUTODBL(NONE)   NOSXM   IL
                      OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)       CHARLEN(500)
                 *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN        1           SUBROUTINE QINP(NSN,NMA,MND,MND1,MARCS2,MIDL,NODE,NDARCS)
                 C...SUBROUTINE MOVES AN ARC FROM SUBSET Q INTO SUBSET P.
ISN        2           INTEGER*4 MIDL(MND),NODE(MND1),NDARCS(MARCS2)
ISN        3           MSTART=MIDL(NSN)+1
ISN        4           MEND=NODE(NSN+1)-1
ISN        5           DO 1 IAC=MSTART,MEND
ISN        6           IF(NDARCS(IAC).EQ.NMA)THEN
ISN        7           ND1=NDARCS(MSTART)
ISN        8           NDARCS(MSTART)=NMA
ISN        9           NDARCS(IAC)=ND1
ISN       10           MIDL(NSN)=MSTART
ISN       11           RETURN
ISN       12           ELSE
ISN       13           ENDIF
ISN       14     1     CONTINUE
ISN       15           RETURN
ISN       16           END
*STATISTICS*   SOURCE STATEMENTS = 16, PROGRAM SIZE = 1260 BYTES, PROGRAM NAME = QINP  .    PAGE:  109.
*STATISTICS*    NO DIAGNOSTICS GENERATED.
**QINP** END OF COMPILATION 45 ******
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:43:21                     PAGE:    55
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE   TERM   OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
                    NOSYM NORENT   SDUMP AUTODBL(NONE)   NOSXM   IL
                     OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)        CHARLEN(500)
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN      1            SUBROUTINE KILTER(MND,MND1,MARCS,MARCS2,ARCUB,AFLOW,VNODE,R,NSETS,
                     +NSTS1,NTAIL,NHEAD,NLOWST,NHIST,MIDP,NODP,NDARCS,JNODE,NSETA,
                     +NEXNOD,PANC,PACOST,CHMAP,LABEL,INLOC,SETMAP,INFES,*)
                C...SUBROUTINE GENERATES AN OPTIMUM CIRCULATION IN A NETWORK
                C...PREVIOUSLY REDEFINED BY THE SUBROUTINE  PSARCS.
ISN      2            INTEGER*4 NTAIL(MARCS2),NHEAD(MARCS2),NODP(NSTS1),INLOC(MND),
                     +MIDP(NSETS),JNODE(MARCS2),NDARCS(MARCS2),PACOST(MARCS2),
                     +NCHA(321),NLOWST(MND),NHIST(MND),NEXNOD(NSETS),LABEL(MND),
                     +SETMAP(NSETS),NSETA(MARCS),CHMAP(321,5),PACCHG,PACOS1
ISN      3            REAL*8 AFLOW(MARCS),PANC(MARCS2),FIP,FLWCHG,PANC1
ISN      4            REAL*8 ARCUB(MARCS),VNODE(MND),R
                C...ENTER THE MAIN LOOP. EXAMINE THE KILTER STATUS OF EACH ARC
                C...AND ITS MIRROR AND DETERMINE THE DESIRED CHANGE.
ISN      5            DO 20 M=1,MARCS
ISN      6            N=M+MARCS
ISN      7            DO 20 KARC=1,2
ISN      8            IF(KARC.EQ.2)THEN
ISN      9            IM=N
ISN     10            IMIR=M
ISN     11            ELSE
ISN     12            IM=M
ISN     13            IMIR=N
ISN     14            ENDIF
ISN     15      22    IF(PANC(IM).LT.-R)THEN
ISN     16            IT=NTAIL(IM)
ISN     17            IS=NHEAD(IM)
ISN     18            KARCS=IMIR
ISN     19            ELSE IF(PANC(IM).GT.R.AND.PACOST(IM).LT.0)THEN
ISN     20            IT=NHEAD(IM)
ISN     21            IS=NTAIL(IM)
ISN     22            KARCS=IM
ISN     23            ELSE IF(PANC(IMIR).LT.-R)THEN
ISN     24            IT=NHEAD(IM)
ISN     25            IS=NTAIL(IM)
ISN     26            KARCS=IM
ISN     27            ELSE
ISN     28            GO TO 20
ISN     29            ENDIF
                C...ENTER LABELING SECTION. INITIALIZE PARAMETERS.
ISN     30            NLN=1
ISN     31            NS=0
ISN     32            NLNCRS=1
ISN     33            LABEL(IT)=KARCS
ISN     34            INLOC(NLN)=IT
ISN     35            NNLS1=1
```

```
                  *....*...1.........2.........3.........4.........5.........6.........7.*.......8
                  C...ENTER LABELING LOOP. EXIT LOOP TO EITHER BREAKTHROUGH OR
                  C...POTENTIAL CHANGE SECTION.
ISN        36     30    CALL LBLRTN(IT,IS,MARCS2,MND,MND1,NNLS1,NLN,NNLS,NEXNOD,
                       +NSETS,NSTS1,NODP,MIDP,NDARCS,NLOWST,NHIST,LABEL,INLOC,NHEAD,*73)
                  C...BREAKTHROUGH SECTION ...
                  C...DETERMINE THE AMOUNT OF PERMISSIBLE FLOW CHANGE.
ISN        37     51    FLWCHG=1.D08
ISN        38           NSN=IS
ISN        39           DO 59 JAC=1,MND
ISN        40           NPA=LABEL(NSN)
ISN        41           IF(NPA.GT.MARCS)THEN
ISN        42           NMA=NPA-MARCS
ISN        43           ELSE
ISN        44           NMA=NPA+MARCS
ISN        45           ENDIF
ISN        46           IF(PACOST(NPA).LE.0)THEN
ISN        47           FIP=PANC(NPA)
ISN        48           VNODE(JAC)=FIP
ISN        49           ELSE
ISN        50           FIP=-PANC(NMA)
ISN        51           VNODE(JAC)=FIP
ISN        52           ENDIF
ISN        53           FLWCHG=DMIN1(FLWCHG,FIP)
ISN        54           NSN=NHEAD(NMA)
ISN        55           IF(NSN.EQ.IS)GO TO 55
ISN        56     59    CONTINUE
                  C...CHANGE THE FLOW ALONG THE FLOW AUGMENTING PATH.
ISN        57     55    DO 61 JAC=1,MND
ISN        58           NPA=LABEL(NSN)
ISN        59           IF(NPA.GT.MARCS)THEN
ISN        60           NMA=NPA-MARCS
ISN        61           ELSE
ISN        62           NMA=NPA+MARCS
ISN        63           ENDIF
ISN        64           PANC1=PANC(NMA)
ISN        65           PANC(NPA)=PANC(NPA)-FLWCHG
ISN        66           PANC(NMA)=PANC(NMA)+FLWCHG
ISN        67           IF(PACOST(NMA).NE.0)GO TO 65
ISN        68           IF(PANC1.GT.R.OR.PANC(NMA).LE.R)GO TO 65
ISN        69           CALL QINP(NSN,NMA,MARCS2,MARCS,NSETS,NSTS1,NSETA,NEXNOD,
                       +SETMAP,CHMAP,MIDP,NODP,NDARCS)
ISN        70     65    IF(VNODE(JAC).EQ.FLWCHG)
                       +CALL PINQ(NHEAD(NMA),NPA,MARCS2,MARCS,NSETS,NSTS1,NSETA,NEXNOD,
                       +SETMAP,CHMAP,MIDP,NODP,NDARCS)
ISN        72           NSN=NTAIL(NPA)
ISN        73           IF(NSN.EQ.IS)THEN
                  C...DELETE LABELS AND CHECK THE KILTER STATUS OF AN ARC.
ISN        74           DO 71 I=1,NLN
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:43:21      NAME: KILTER  PAGE:   57
                      *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN       75    71     LABEL(INLOC(I))=0
ISN       76           GO TO 22
ISN       77           ELSE
ISN       78           ENDIF
ISN       79    61     CONTINUE
                C...POTENTIAL CHANGE SECTION.
ISN       80    73     NSLPCH=NS
ISN       81           NRLPCH=NR
ISN       82           PACCHG=1E09
ISN       83           NBRK=0
ISN       84           NS=0
ISN       85           NR=MARCS2
ISN       86           IF(NSLPCH.EQ.0)GO TO 75
ISN       87           IF(NRLPCH.EQ.MARCS2)GO TO 74
                C...NR SET RECYCLING SECTION.
ISN       88           DO 76 J=MARCS2,NRLPCH+1,-1
ISN       89           JL=JNODE(J)
ISN       90           JNEXT=NHEAD(JL)
ISN       91           IF(LABEL(JNEXT).NE.0)GO TO 76
ISN       92           JNODE(NR)=JL
ISN       93           NR=NR-1
ISN       94    76     CONTINUE
                C...NS SET RECYCLING SECTION.
ISN       95    74     DO 77 J=1,NSLPCH
ISN       96           JL=JNODE(J)
ISN       97           JNEXT=NHEAD(JL)
ISN       98           IF(LABEL(JNEXT).NE.0)GO TO 77
ISN       99           IF(PACOST(JL).GT.0)THEN
ISN      100           NS=NS+1
ISN      101           JNODE(NS)=JL
ISN      102           PACCHG=MIN(PACCHG,PACOST(JL))
ISN      103           ELSE
ISN      104           JNODE(NR)=JL
ISN      105           NR=NR-1
ISN      106           ENDIF
ISN      107    77     CONTINUE
ISN      108    75     IF(NLNCRS.GT.NLN)GO TO 90
                C...DETERMINE THE MIN MARGINAL COST CHANGE.
ISN      109           DO 80 ND=NLNCRS,NLN
ISN      110           I=INLOC(ND)
ISN      111           MSTART=NLOWST(I)
ISN      112           MEND=NHIST(I)
ISN      113           DO 83 MSET=MSTART,MEND
ISN      114           IF(LABEL(NEXNOD(MSET)).NE.0)GO TO 83
ISN      115           NSTART=MIDP(MSET)+1
ISN      116           NEND=NODP(MSET+1)-1
ISN      117           DO 81 JP=NSTART,NEND
ISN      118           NA=NDARCS(JP)
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:43:21       NAME: KILTER  PAGE:    58
                    *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN       119             IF(PANC(NA).LT.-R.OR.PACOST(NA).LE.0)THEN
ISN       120             JNODE(NR)=NA
ISN       121             NR=NR-1
ISN       122             ELSE
ISN       123             NS=NS+1
ISN       124             JNODE(NS)=NA
ISN       125             PACCHG=MIN(PACCHG,PACOST(NA))
ISN       126             ENDIF
ISN       127       81    CONTINUE
ISN       128       83    CONTINUE
ISN       129       80    CONTINUE
ISN       130       90    NLNCRS=NLN+1
                    C...DETERMINE IF PROBLEM IS INFEASIBILE.
ISN       131             IF(NS.EQ.0)THEN
ISN       132             INFES=1
ISN       133             GO TO 33
ISN       134             ELSE
ISN       135             ENDIF
                    C...CHANGE OF MARGINAL COST FOR SET S.
ISN       136             DO 91 I=1,NS
ISN       137             NPA=JNODE(I)
ISN       138             IF(NPA.GT.MARCS)THEN
ISN       139             NMA=NPA-MARCS
ISN       140             ELSE
ISN       141             NMA=NPA+MARCS
ISN       142             ENDIF
ISN       143             PACOST(NPA)=PACOST(NPA)-PACCHG
ISN       144             PACOST(NMA)=-PACOST(NPA)
ISN       145             IF(PACOST(NPA).NE.0.OR.PANC(NPA).LE.R)GO TO 91
ISN       146             CALL QINP(NTAIL(NPA),NPA,MARCS2,MARCS,NSETS,NSTS1,NSETA,NEXNOD,
                         +SETMAP,CHMAP,MIDP,NODP,NDARCS)
ISN       147             NLBA=NHEAD(NPA)
ISN       148             IF(LABEL(NLBA).NE.0)GO TO 91
ISN       149             LABEL(NLBA)=NPA
ISN       150             NLN=NLN+1
ISN       151             INLOC(NLN)=NLBA
ISN       152             IF(NLBA.EQ.IS)NBRK=1
ISN       154       91    CONTINUE
                    C...CHANGE OF MARGINAL COST FOR SET R.
ISN       155             IF(NR.EQ.MARCS2)GO TO 99
ISN       156             DO 92 I=NR+1,MARCS2
ISN       157             NPA=JNODE(I)
ISN       158             IF(NPA.GT.MARCS)THEN
ISN       159             NMA=NPA-MARCS
ISN       160             ELSE
ISN       161             NMA=NPA+MARCS
ISN       162             ENDIF
ISN       163             PACOS1=PACOST(NPA)
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:43:21       NAME: KILTER  PAGE:   59
                         *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN      164            PACOST(NPA)=PACOS1-PACCHG
ISN      165            PACOST(NMA)=-PACOST(NPA)
ISN      166            IF(PACOS1.LT.0.OR.PACOST(NPA).GE.0)GO TO 92
ISN      167            IF(PANC(NPA).LT.-R.OR.PANC(NPA).GT.R)GO TO 92
ISN      168            CALL PINQ(NHEAD(NPA),NMA,MARCS2,MARCS,NSETS,NSTS1,NSETA,NEXNOD,
                       +SETMAP,CHMAP,MIDP,NODP,NDARCS)
ISN      169       92   CONTINUE
                   C...OUT OF KILTER CHECK.
ISN      170       99   IF(PANC(IM).LT.-R.OR.PANC(IMIR).LT.-R)THEN
                   C...CHECK BREAKTHROUGH AS A RESULT OF POTENTIAL CHANGE.
ISN      171            IF(NBRK.EQ.1)GO TO 51
ISN      172            NNLS1=NNLS
ISN      173            IF(NNLS1.EQ.NLN)GO TO 73
ISN      174            NNLS1=NNLS1+1
ISN      175            GO TO 30
ISN      176            ELSE IF(PANC(IM).GT.R.AND.PACOST(IM).NE.0)THEN
ISN      177            IF(NBRK.EQ.1)GO TO 51
ISN      178            NNLS1=NNLS
ISN      179            IF(NNLS1.EQ.NLN)GO TO 73
ISN      180            NNLS1=NNLS1+1
ISN      181            GO TO 30
ISN      182            ELSE
                   C...ERASE LABELS TO PREPARE FOR THE NEXT ARC.
ISN      183            DO 98 I=1,NLN
ISN      184       98   LABEL(INLOC(I))=0
ISN      185            ENDIF
ISN      186       20   CONTINUE
                   C...CONVERT FLOWS BACK INTO ACTUAL VALUES.
ISN      187       33   DO 32 M=1,MARCS
ISN      188            IF(PANC(M).LT.R)PANC(M)=0.0
ISN      190            AFLOW(M)=ARCUB(M)-PANC(M)
ISN      191       32   CONTINUE
ISN      192            IF(INFES.EQ.1)RETURN 1
ISN      194            RETURN
ISN      195            END
*STATISTICS*    SOURCE STATEMENTS = 191, PROGRAM SIZE = 8740 BYTES, PROGRAM NAME = KILTER    PAGE:   55.
*STATISTICS*     NO DIAGNOSTICS GENERATED.
**KILTER** END OF COMPILATION 26 ******
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:43:19                       PAGE:    22
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE   TERM   OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
                    NOSYM NORENT   SDUMP AUTODBL(NONE)   NOSXM   IL
                      OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)      CHARLEN(500)
                  *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN        1            SUBROUTINE ARRAYS(MARCS,MND,MARCS2,MND1,R,NSETS,NSTS1,
                       +NTAIL,JNODE,PANC,PACOST,NODARC,ARCUB,AFLOW,ARCLB,
                       +NCHA,SETMAP,MIDP,NODP,NDARCS,NLOWST,NHIST)
                  C...SUBROUTINE TRANSFORMS THE ORIGINAL CAPACITATED NETWORK INTO
                  C..." PSEUDO-ARCS " NETWORK AS REQUIRED BY THE REFORMULATED OUT-
                  C...OF-KILTER ROUTINE.
ISN        2            INTEGER*4 NTAIL(MARCS2),NODP(NSTS1),SETMAP(NSETS),
                       +MIDP(NSETS),JNODE(MARCS2),NDARCS(MARCS2),PACOST(MARCS2),
                       +NCHA(321),NODARC(NSETS),NLOWST(MND),NHIST(MND)
ISN        3            REAL*8 AFLOW(MARCS),PANC(MARCS2)
ISN        4            REAL*8 ARCUB(MARCS),R,ARCLB(MARCS)
                  C...THIS SECTION CREATES PSEUDO ARCS.
ISN        5            DO 7 M=1,MARCS
ISN        6            N=M+MARCS
ISN        7            PANC(M)=ARCUB(M)-AFLOW(M)
ISN        8            PANC(N)=AFLOW(M)-ARCLB(M)
ISN        9      7     CONTINUE
ISN       10            DO 8 K=1,NSETS
ISN       11            JNODE(K)=NODP(K)
ISN       12            MIDP(K)=NODP(K+1)-1
ISN       13      8     CONTINUE
ISN       14            JNODE(NSTS1)=MARCS2+1
                  C...THIS SECTION CREATES ARRAYS NDARCS ,MIDP AND NODP. NDARCS IS
                  C...A SET OF OUTGOING ARCS FOR EACH NODE, WHILE MIDP AND NODP ARE
                  C...VECTORS THAT SPLIT NDARCS SUB-SETS (ASSOCIATED WITH ARC SETS
                  C...CONNECTED TO EACH NODE) INTO LABEL ELIGIBLE AND INELIGIBLE ARCS.
ISN       15            DO 9 I=1,MND
ISN       16            NSTRT=NLOWST(I)
ISN       17            NEND=NHIST(I)
ISN       18            DO 11 L=NSTRT,NEND
ISN       19            LMIN=NODARC(L)
ISN       20            LMAX=LMIN+NCHA(SETMAP(L))-1
ISN       21            DO 6 MM=LMIN,LMAX
ISN       22            J=NTAIL(MM)
ISN       23            IF(I.NE.J)THEN
ISN       24            MB=MM
ISN       25            M=MM+MARCS
ISN       26            ELSE
ISN       27            MB=MM+MARCS
ISN       28            M=MM
ISN       29            ENDIF
ISN       30            IF(PACOST(M).GT.0.AND.PANC(MB).LT.-R)THEN
ISN       31            IC=JNODE(L)
ISN       32            JNODE(L)=IC+1
ISN       33            ELSE IF(PACOST(M).LE.0.AND.PANC(M).GT.R)THEN
```

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

NL-339 (r. 88/04)

Canada

UNIVERSITY OF ALBERTA

An Improved Out-of-kilter Algorithm applied to the Water Resources Management Model of Alberta Enviroment

by

Ryan Ilich

©

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE

DEPARTMENT OF CIVIL ENGINEERING

EDMONTON, ALBERTA

Fall 1988

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-45613-2

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR	Ryan Ilich
TITLE OF THESIS	An Improved Out-of-kilter Algorithm applied to the Water Resources Management Model of Alberta Enviroment

DEGREE FOR WHICH THESIS WAS PRESENTED MASTER OF SCIENCE

YEAR THIS DEGREE GRANTED Fall 1988

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(SIGNED) Ryan Ilich

PERMANENT ADDRESS:

19 BERNARD DR. NW.
CALGARY, ALBERTA
T3K 2B4

DATED August 26, 1988.

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled An Improved Out-of-kilter Algorithm applied to the Water Resources Management Model of Alberta Enviroment submitted by Ryan Ilich in partial fulfilment of the requirements for the degree of MASTER OF SCIENCE.

....

Supervisor

....

....

Date. Aug. 25, 1988

ABSTRACT

The Water Resources Management Model (WRMM) of Alberta Environment is a large computer program developed to help the overall water resources planning of Alberta Environment. The WRMM uses a special type of linear programming routine, known as the out-of-kilter (OKA) algorithm to derive an optimal allocation of water in the basin.

The OKA algorithm was originally developed by D. Fulkerson and L. Ford in 1961 as a special type of linear programming routine for network optimization problems. In 1974, R. Barr, D. Klingman and F. Glover presented a reformulated OKA which was claimed to be three to ten times more efficient than the OKA of Ford and Fulkerson.

Two different codes based on the reformulated algorithm were developed in overall compliance with the WRMM. In one of them, Barr's idea of partitioning was slightly modified with further sub-partition of arc sets associated with each node, rather than having only one partition of all arcs associated with a node.

As previously reported by Barr, the crucial parameter for computational improvements was the number of nodes, rather than the number of arcs in the network. The best results that they achieved were on networks with over 500 nodes. According to their report, networks with less than 100 nodes were solved only 2 to 3 times faster. The new version of the WRMM indicates similar improvement for networks with less than 30 nodes, and the total cpu time is

15% to 25% less than in the case of the old version. Since the maximum number of nodes in the WRMM is limited to 102, it didn't seem logical to expect dramatic changes for larger networks. However, the improvements of the cpu time for the largest networks surpassed the expectations based on Barr's paper. The South Saskatchewan ten year simulation run was executed in 30.03 seconds total, while the old version needed 87.5 seconds for the same job. For this particular run, the old OKA subroutines took 73.47 seconds, or roughly 84% of the total CPU time.

ACKNOWLEDGEMENTS

I would like to express my thanks to the people and organizations who have, in various ways, contributed towards the successful completion of this project.

The major initiator and sponsor of this project was the Planning Division of Alberta Environment, Calgary. Financial assistance was also received from Nanuk Engineering and Development ltd., Calgary, and the National Research Council of Canada. Their help is most appreciated.

Among individuals, I am most indebted to Alan Wright of Alberta Environment, who awarded me with outstanding confidence and support from the earliest stages of this project.

I am also thankful to Mike Thompson, Russ Lewis and Vincent Chiu of Nanuk Engineering and Development ltd. for their support and suggestions during the course of this project.

Finally, I would like to thank the Department of Civil Engineering for accepting this project as a thesis topic, and particularly Dr. Peter Steffler, my supervisor, for his effort involved in dealing with this thesis topic.

PREFACE

The main purpose of this project was to reduce the computation time of the Water Resources Management Model (WRMM) of Alberta Environment.

The WRMM is a deterministic surface water allocation model written in FORTRAN 77. It derives an optimal allocation policy within a modelled region with respect to given input (inflows, water demand and management priorities among the users). Originally developed for the South Saskatchewan River Basin Planning Program, the WRMM can model any river basin for a very large number of prescribed management priorities.

For each component of the system (natural channel, reservoir, etc.) a finite number of flow zones is specified. An ideal zone for each element is assigned a penalty of zero. Other zones below and above the ideal zone have penalties specified by the user such that the further a zone is from the ideal zone, the greater is its penalty. This zoning concept was originally developed by the U.S. Army Corps of Engineers [35], and implemented in Canada by Acres Consulting Services [38].

There are sixty three subroutines in the WRMM altogether; seven of them are optimization subroutines based on the Out-of-Kilter algorithm (OKA) of Ford and Fulkerson, [36] and they account for the major portion of the cpu time.

The Ford-Fulkerson algorithm was the fastest linear programming routine for solving minimum cost flow problems in networks at the time it was developed. However, the computational cost of using WRMM was still prohibitive. In 1974, Barr, Glover and Klingman [52] reformulated the OKA, splitting each arc in the network into two "pseudo arcs". New primal and dual variables were associated with each pseudo arc. They also introduced three more arrays to keep track of the label eligibility of arcs. These changes were aimed at simplifying the labelling and the potential change procedures, which account for most of the computational effort required by the OKA.

Other properties of Ford-Fulkerson's original algorithm are still maintained in the new version. The main feature of both versions is to bring each arc of the network, one by one, into one of the "good conditions", or "in-kilter" states. If all arcs can be brought in-kilter, an optimal solution is found; if not, the problem is declared infeasible. The kilter state of an arc is defined by the values of its primal and dual variables.

The purpose of this project is summarized as follows:

a) Develop subroutines based on the reformulated algorithm in FORTRAN 77 with the same variable names as in the WRMM and test them outside the WRMM.

b) Replace the existing subroutines in the WRMM with the new subroutines, taking care that:

- Original arrays related to the internal network of the WRMM are preserved.
- Existing output options of the WRMM are still available.

This means that the internal network used in the old version is stored, since one of the output options includes a printout of the entire internal network.

c) Evaluate performance of the new method with respect to accuracy and execution efficiency. Also, examine the possibilities of further improvements of the new method.

Following the above criteria results in minimal changes of the existing WRMM documentation. This is important, since this model is currently used by two provincial governments (Saskatchewan and Alberta) and the Federal Government of Canada.

Table of Contents

Chapter Page

1. WATER RESOURCES MODELLING 1

1.1 Introduction 1

1.2 Types of Water Resources Models 3

1.2.1 General Model Structure 3

1.2.2 Optimization in Water Resources Modelling ..4

1.2.3 The Origin of Linear Decision Rule and the Zoning Concept 7

2. BRIEF DESCRIPTION OF THE WRMM 15

2.1 General Features 15

2.2 Water Allocation Policies 16

2.3 Interreservoir and Cumulative Priorities 19

2.4 General Model Structure 20

2.5 Input Data 25

2.6 Flow Network Organization 28

2.7 Numerical Example 36

2.8 Final Remarks 41

2.8.1 Natural Channel Flows 41

2.8.2 Infeasible Solutions 42

2.8.3 FORTRAN 77 vs Other High Level Programming Languages 43

3. THE OUT-OF-KILTER ALGORITHM OF FORD AND FULKERSON ...45

3.1 Introduction to Linear Programming 45

3.1.1 Definition of a Linear Program 45

3.1.2 The Concept of Duality 47

3.2 Introduction to Network Minimum Cost Flow Problem 51

3.3 The Out-of-Kilter Algorithm 53

3.3.1 Theoretical Basis of the Algorithm 53

3.3.2 General Description of the Method 58

4. AN IMPROVED OKA OF BARR, GLOVER AND KLINGMAN 68

4.1 Numerical Effort of the Labelling and Potential Change Procedure 68

4.2 Description of the Improved OKA of Barr et al. ..70

4.2.1 Pseudo Arcs and Pseudo Network 70

4.2.2 Net Capacity and Marginal Cost 71

4.2.3 New Labelling Procedure 72

4.2.4 Potential Change Procedure 78

4.3 Flowchart of Barr's Algorithm 82

5. DETAILED DESCRIPTION OF THE NEW OPTIMIZATION SUBPROGRAM IN THE WRMM 87

5.1 General Features 87

5.2 Numerical Example 90

5.3 Specific Features of the WRMM Internal Network .107

6. CHANGES IN THE NEW VERSION OF WRMM 111

6.1 Changes in Memory Requirements 111

6.2 Changes in the Main Program and Other Subroutines 112

6.3 Comparison of Results 115

6.4 Comparison of Execution Times 116

6.5 FURTHER ENHANCEMENTS OF BARR'S ALGORITHM 119

6.6 FORTRAN Data Structure Limitations 121

6.6.1 Linked Lists vs Physically Ordered Lists .121

6.6.2 PASCAL Data Structures 123

6.6.3 An Example of a Linked List Database Model for Barr's Algorithm 124

7. CONCLUSION 132

REFERENCES 135

APPENDIX I 142

APPENDIX II 149

APPENDIX III 150

List of Tables

Table Page

2.1 Component vs modelled processes........................23

5.1 Primal and dual values and P and Q sets..............99

5.2 Primal and dual values and P and Q sets.............100

5.3 Primal and dual values and P and Q sets.............101

5.4 Primal and dual values and P and Q sets.............102

5.5 Primal and dual values and P and Q sets.............103

5.6 Primal and dual values and P and Q sets.............104

5.7 Primal and dual values and P and Q sets.............105

5.8 Primal and dual values and P and Q sets.............106

5.9 Primal and dual values and P and Q sets.............107

6.1 CPU TIME FOR BOWOP85 SIMULATION RUM.................117

6.2 CPU TIMES FOR SOUTH SASKATCHEWAN SIMULATION RUN......118

List of Figures

Figure Page

2.1 Example set of separating policies....................18

2.2 Flow zones..29

2.3 Flow arcs...29

2.4 Example external network..........................32

2.5 Example internal network..........................33

2.6 Example reservoir zones...........................39

2.7 Example elevation vs storage curve................39

3.1 Labelling procedure - non-breakthrough............63

3.2 Labelling procedure - breakthrough................63

5.1 Simple circulatory network........................91

5.2 Pseudo network....................................92

6.1 ARC RECORD.......................................124

6.2 NODE RECORDS.....................................125

1. WATER RESOURCES MODELLING

1.1 Introduction

The field of water resources management has shown a remarkable growth over the past few decades. The problem of water resources management (in its most general form [1]) arises from the fact that water is usually available in location, quantities, qualities and at times which are not most suitable to the users. This has been the case even in the earliest stages of human development. However, the amount of water consumption per capita in today's industrialized world has exceeded that of early primitive societies by hundreds of times. This has placed considerable requirements on present and future water resources management.

A number of major issues related to energy, food and environment are closely associated with water resources management. Listed below are some of the activities which are incorporated in water resources management:

- Municipal and industrial water supply
- Irrigation and drainage
- Flood control
- Hydro power
- Fish and wildlife
- Navigation
- Recreation

- Watershed management
- Pollution control

Due to the complexity of all the issues involved, a number of relatively recent problem solving techniques (usually identified under terms like "operational research", or "systems analysis"), have become valuable tools in water resources management. Most of these techniques have in common a mathematical model of the real world system. Modern "water resources modelling" is based on extensive application of these techniques. Some of the characteristics of water resources systems are:

- Conflict among different users
- Increasing areal spread
- Dynamic character of most of the parameters
- Stochastic character of the input parameters
- Environmental impacts
- Specific economic nature
- Socio-political nature

Water resources models still maintain a very limited scope in dealing with these issues. They are able to tackle, with varying success, only one issue at a time, subject to the limitations of the model and to the experience of the modeler. In spite of these limitations, modelling is viewed as a useful tool in modern water resources management and efforts devoted to development and implementation of various models have been growing in recent years.

1.2 Types of Water Resources Models

There are many types of models in water resources management today. The following discussion is restricted to those known as the computer simulation models.

1.2.1 General Model Structure

Similar to any other mathematical model, water resources models consist of three parts:

1) Input variables
2) Functions which link input to output
3) Output

This holds for all models. In a broader sense, modelling can be viewed as a search for 1), 2) or 3), depending on which of them is unknown.

Mathematical functions in water resources models are derived from basic natural principles, such as the conservation of mass or conservation of energy. However, practical straightforward application of these principles is often impossible, mainly due to the difficulty of isolating the boundary of the control volume in particular cases. The rainfall-runoff models are a good illustration of this difficulty. The first model of this kind dates back from the 1930s [2]. It was based on simple measurement of input and output (actual rainfall and runoff on a particular watershed) and establishing a direct correlation between these two, with calibration of as few parameters as possible. Other models aimed to solve the same problem have followed since, based on the queueing theory [3], Markov processes [4], digital simulation [5], [6], etc. However, none of them achieved greater reliability than the original unit hydrograph. The reason for this is that all of these models were based on calibration of parameters in such a way that the modelled input and output are same as those already measured in nature.

There are many different classifications of water resources models. Some of the most common are:

1) deterministic and stochastic
2) steady state and unsteady state

These distinctions are based on the way in which the variables are handled by the model.

1.2.2 Optimization in Water Resources Modelling

Models which are used as a tool in studying natural processes like: sediment transport, flow regimes, surface runoff or groundwater flow are called "non-optimizing", or "descriptive" [7]. The need for optimization models stems from the very nature of water resources management: any attempt to change natural flow regimes in order to satisfy human needs has always been followed by examining all the available options and choosing one which seems most preferable.

Basic model structure mentioned earlier is also maintained in case of optimization models. What makes them different from the descriptive models is the existence of a built-in criteria, formulated in the form of "the objective function", on the basis of which a solution is derived by the model. Most of the optimization models are based on mathematical techniques by which the realm of feasible solutions is searched until one (or more) of them are isolated, such that the objective function has reached its extreme value (maximum or minimum, depending on the nature of the problem).

The actual optimization techniques applied in water resources management are not different from those applied in other areas [8], [9]. Most of them are based on linear or non-linear programming. Some of the most recent optimization algorithms (known as "genetic algorithms" [10]) have considerable potential for successfully replacing existing algorithms.

Practical use of optimization models in general was greatly encouraged by economic competition, and that the first successful optimization models became well known because they helped the maximization of profit (or minimization of cost) in the particular industrial problems to which they were applied. In that respect, the 1950's marked the beginning of the whole revolution which involved rapid development of optimization algorithms and their applications in all walks of life. Water resources management was not exempted from this trend. The common problem related to water resources management models is the complexity of water resources systems, which has the following implications:

1) Very high computational cost
2) Validity of the objective function

For example, modelling a river basin with fifty components (e.g. irrigation fields, diversion structures, reservoirs, etc) where each component may take one of three possible trial values (i.e. storage or flow capacities) would involve examining the total number of 3^{50} combinations! Many modelling concepts were abandoned because of this limitation. As a result of this, various modelling concepts [11] were designed in order to reduce the computational effort.

The most frequent concept applied in constructing objective functions in water resources management models so far has been based on economic evaluation. A typical example is the calculation of the net benefit function, evaluated as gross benefit (incurred by adding additional stimulus to the economy through increasing the availability of water) minus the costs of related water resources projects. Numerous studies were completed based on this principle [12], [13], [14], aimed at evaluating planning alternatives such as: viability of interbasin transfers and optimizing reservoir sizes or reservoir releases.

The major difficulty in all these studies is that many aspects of water resources management (e.g. environment, recreation, wildlife, etc.) are not easy to quantify in monetary terms. This has resulted in a search for other types of objective functions, which wouldn't be necessarily based on direct economic input, but which would still capture different model responses that incorporate engineering judgement.

1.2.3 The Origin of Linear Decision Rule and the Zoning Concept

The further discussion is limited to models which are aimed to derive optimal reservoir management policies. Reservoirs are a major tool in changing the natural flow regime, which makes this issue the central part of any water resources management model. Listed below is a brief review of the developments in reservoir optimization models within the past few decades. A general queuing theory approach was adopted by Thomas and Watermeyer [15] in order to determine optimal reservoir release using a linear programming model. It was assumed that inflows were independent random events with a known probability distribution.

Dynamic programing (DP) was also used in optimizing reservoir releases. A stochastic[1] DP approach was used by Little [16] and Buras [17] in various single reservoir models. Hall [18] applied DP for the design of a single multipurpose reservoir, Young [19], [20] combined deterministic DP and hydrologic simulation and Harboe [21] used a DP approach for optimizing a long term operating policy for a single multipurpose reservoir. Most of the early studies focused on single reservoir models. Parikh [22] combined a DP approach for a single reservoir with the decomposition principle of linear programming to link the reservoirs in the system. This approach was later extended by Buras [23] and Hall [24].

---

[1] Flow changes are considered stochastic since the expectancy of the flow valus varies with time on an annual basis.

All of these studies had one conclusion in common: the optimization of reservoir releases was found to be computationally expensive. For example, a single reservoir model by Gablinger and Loucks [25] contained approximately 2,000 equations and 15,000 variables and it required two hours of IBM 360/65 CPU time . Even worse were multireservoir models: a three reservoir system was run on the model of Roefs and Bodin [26], and after twenty hours of IBM 360/50 CPU time it still hadn't reached an optimal solution.

In 1969, Revelle et al [27] adapted the use of the "linear decision rule", which was originally proposed by Charnes [28]. Reservoir release for one time interval was related to the storage at the end of the previous time period. The change of storage for one particular time interval was derived by the model. The advantages of this approach are in easy linear programming formulation as well as in a convenient way of defining the objective function. This approach has been applied in numerous studies since its inception [29], [30], [31], although it was also subject to criticism [32], [33], which was focused on questioning the flexibility of linear decision rules in reservoir management.

Some of the advantages of this approach were:

1) significant reduction of the computational cost
2) flexibility of having an option to vary the length of time intervals
3) a convenient representation of the prescribed priorities among the users in the form of the penalty point system

The penalty point system has introduced a more flexible objective function, especially in areas like water recreation or fish and wildlife, which are virtually impossible to quantify economically. Moreover, it was designed to represent a priority policy in place for a particular region being modelled.

To run a model based on this concept the following information is needed:

1) definition of current state of the system (channel flows and reservoir levels)
2) prediction of natural inflows into the system as well as losses due to evapotranspiration and seepage
3) allocation priority for all the users in the system, contained in the penalty point system

Based on the current state and prediction of net inflows for the next time interval, the model derives an optimal policy for one time interval. This optimal solution is the starting point for the next time interval, thus creating a series of optimal solutions for the whole simulation period. Prediction of net inflows can represent the actual data recorded in the past, or if this is not available, flows could be derived synthetically using various runoff models.

The application of the linear decision rule in reservoir optimization is based on specifying the rule curve for each reservoir, i.e. ideal storage versus time relationship. Modellers can exercise flexibilty in defining this relationship, as long as the physical constraints involved are not ignored (e.g. maximum or minimum elevation for a particular reservoir being modelled). If at any point in time natural supply is not sufficient to meet the ideal requirements (during a simulation run), the model derives release in reservoir with the lowest priority first.

This is the principle framework of using the linear decision rule in a multireservoir model. It was significantly enhanced by the development of the zoning concept [34], [35], which allowed for inclusion of all components (natural channels, irrigation, municipal, hydro, etc.) with often conflicting demands into modelling. The ultimate goal of such optimization models is to define a strategy which resolves the conficts to the best possible extent. Naturally, this will only hold provided that all parties involved had agreed on operating priorities prior to program execution. The zoning concept is based on dividing a reservoir storage into a number of storage zones, which were named in some studies (starting from top to bottom) as the spill zone, the flood control zone, conservation zone, buffer zone and inactive zone. The actual number of zones is arbitrary, but the common feature is that the top of the conservation zone is the ideal level, and any other storage which deviates from this is "penalized" according to the penalties assigned to each zone. The same concept was applied to all other components of the system. The advantages of this approach could be summarized by the following:

1) It is possible to examine the operation of the system for a number of different operational policies.
2) The modelled region is extremely flexible. It can vary from small districts to the largest river basin.
3) Top and bottom limit of every zone can be specified with a different value for each time interval. This has particular importance, since ideal storage (or flows) in components are time dependent.
4) It is possible to capture important operational interdependence of some components by assigning proper penalty values. For example, flows in natural channels depend on current storage of their respective upstream reservoirs. To illustrate this, observe a reservoir and the corresponding downstream channel in some typical situations:
    a) Inflow into the reservoir is such that the storage is increasing above the ideal level and it begins to fill the flood storage zone. The penalty in this zone is set slightly higher than the penalty in the

corresponding zone of the natural channel below the reservoir (the one immediately above the ideal zone). Since the model tends to minimize the total system penalty, it will release as much flow as possible to the natural channel and prevent overly rapid filling of the flood storage zone.

b) Similarly, in dry periods, the tendency will be reversed: instead of keeping release at its maximum, it is reduced to the minimum. Again, penalties of lower reservoir zones are set higher than corresponding natural channel zones. In other words, flow in a downstream channel is first reduced to the bottom of its lower zone, before any reservoir releases are to take place. The modes of operation described under a) and b) are in effect in the operation of the actual real world reservoirs. What is being emphasized here is the simplicity of modelling this feature using the zonal penalty concept.

5) Efficient solution procedure: this modelling approach allows for application of a special linear programing technique, known as the Out-of kilter algorithm (OKA) [36], [37], which has proved to be more efficient in terms of computational time than a standard linear programing technique.

Finally, this modelling approach has some drawbacks as listed below:

1) The application of the OKA is based on the assumptions of a steady state situation for a time interval. In other words, all output flows are given as average flows within time intervals. Consequently, input data (natural inflows) must be converted into the same form. This constitutes a departure from reality, since if a time interval is one month, flows may vary within some months considerably, and yet the model will be able to accept only the averaged values. The solution is in specifying shorter time intervals (e.g. one week), but for large river basins with simulation periods of seventy years or more, this still appears to be too expensive in terms of computational cost.

2) This modelling concept requires extensive hydrologic records in order to be applied successfully to a particular river basin. The data required includes:
   - Historical natural channel flows for as many locations in the basin as possible
   - Assessment of water demands by municipal, hydro, irrigation and other components involved
   - Evaporation and precipitation data
   - Storage vs elevation curves for all existing and potential reservoir sites

In the case when the foregoing information is not available, it has to be generated synthetically, using some of the available techniques. However, the data derived in this way are less reliable than the historically recorded data. Consequently the model output also becomes less reliable. So far, this modelling concept was successfully utilized in a number of studies which were completed for river basins with extensive historical records available [38], [39]. In the case of the Trent river basin study [40], a system consisting of 48 reservoirs was modelled successfully, which was unthinkable in terms of earlier techniques. The developments undertaken by Alberta Environment in 1981 were, with minor modifications, based on the strategy envisaged in the Trent river basin study. This is mainly due to the previously mentioned advantages of this approach, as well as the existence of hydrologic records for the South Saskatchewan River Basin for more than seventy years.

2. BRIEF DESCRIPTION OF THE WRMM

2.1 General Features

The Water Resources Management Model (WRMM) of Alberta Environment [39], [41] is a deterministic surface water allocation model. It was originally developed as part of the South Saskatchewan River Basin Planning Program. This model can be used in two ways:

a) Long term basin planning, which uses historical hydrologic data to represent future conditions.

b) Operational planning, which attempts to derive short term operational policies based on the hydrologic forecast.

Successful application of this model is enhanced by the following:

1) This model can be applied to any river basin. All physical parameters (storage vs elevation curves, hydrographs, precipitation and evaporation data) as well as the river basin topology (junctions, reservoirs etc.) are specified in the input data file and can be easily changed.

2) The modelled region is extremely flexible; it can vary from large river basins interconnected with interbasin transfers to a small irrigation district with its internal canals and storage components.

3) Simple change of operational priorities is an extremely significant flexibility of this model. These are

specified through the penalty point system, which is defined by the user in the input data file. This allows the user to examine easily a number of different operational priorities by only changing the values of some of the penalties in subsequent simulation runs. Since the penalty point system represents the central idea on which the whole model is based, it will be discussed more thoroughly in the following section.

2.2 Water Allocation Policies

As previously mentioned, allocation priorities are modelled using the zoning concept in combination with the penalty point system. Under ideal conditions all demands are satisfied and flows in all components are within their "normal" limits. However, during periods of water surplus or shortage, deviations from the ideal conditions are inevitable. There are two factors which determine the way in which the deviations will occur in the model:

1) values of the penalties associated with each zone
2) boundaries between the zones (i.e. the size of each zone)

Both of these factors are equally important and they act in conjunction with each other. In order to illustrate this concept, a simple example is presented. There are four components modelled in this example: a reservoir, a natural channel, an irrigation and an apportionment channel. The last component is a unique feature of WRMM, which models apportionment agreements between states or provinces.

All components and their respective zones and penalties are shown in Figure 2.1. Penalties are assigned per unit of flow. In the case of reservoirs, the current storage deviation (the difference between the rule curve storage and the current storage for a particular time interval) is divided with the length of the time interval, and this "storage equivalenced" flow is also assigned penalty per unit of flow. Since the overall tendency is to minimize the total system penalty, the deviations will first occur in zones with lowest penalties. In the example in Fig. 2.1. the increasing shortage will cause the following changes (in the order listed):

1) The storage in the reservoir will be brought to the bottom of the first relaxation zone, since this zone has the lowest penalty (1).
2) Before reservoir drawdown into the second relaxation zone, the natural channel flow will reach the bottom of its first relaxation zone.
3) Next, the irrigation demand will not be met 100%. The amount of irrigation consumption will equal the one specified as the bottom of the first relaxation zone.
4) Further decrease in natural supply will result in reservoir storage decrease to the bottom of the second relaxation zone, before the apportionment flows are reduced, and so on.

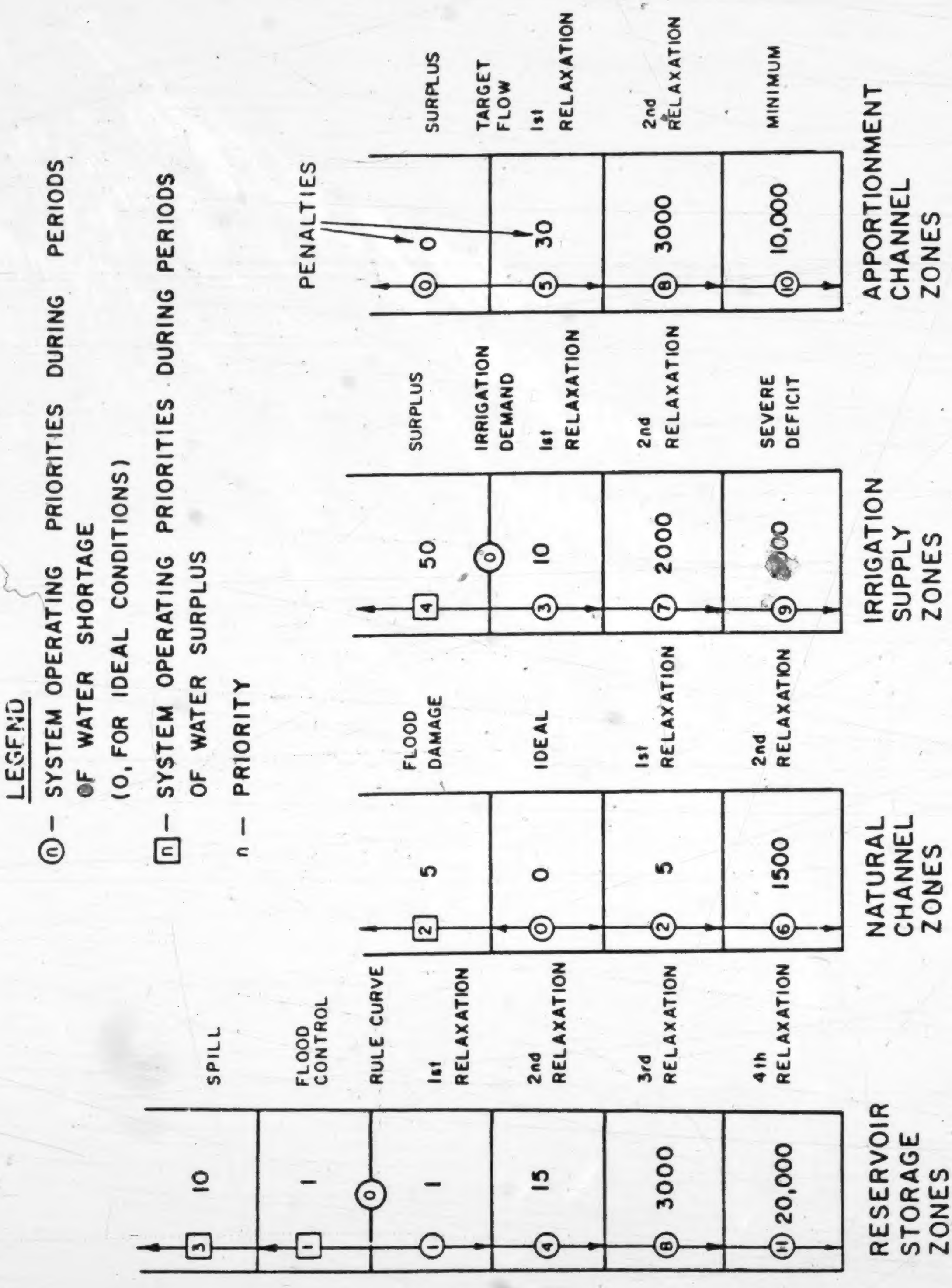

LEGEND
(n) — SYSTEM OPERATING PRIORITIES DURING PERIODS OF WATER SHORTAGE
(0, FOR IDEAL CONDITIONS)
[n] — SYSTEM OPERATING PRIORITIES DURING PERIODS OF WATER SURPLUS
n — PRIORITY
PENALTIES
RESERVOIR STORAGE ZONES
SPILL
FLOOD CONTROL
RULE CURVE
1st RELAXATION
2nd RELAXATION
3rd RELAXATION
4th RELAXATION
3
10
1
1
0
1
1
4
15
8
3000
11
20,000
NATURAL CHANNEL ZONES
FLOOD DAMAGE
IDEAL
1st RELAXATION
2nd RELAXATION
2
5
0
0
2
5
6
1500
IRRIGATION SUPPLY ZONES
SURPLUS
IRRIGATION DEMAND
1st RELAXATION
2nd RELAXATION
SEVERE DEFICIT
4
50
0
3
10
7
2000
9
APPORTIONMENT CHANNEL ZONES
SURPLUS
TARGET FLOW
1st RELAXATION
2nd RELAXATION
MINIMUM
0
0
5
30
8
3000
10
10,000

fig 2.1 Example set of separating policies

System operating priorities during periods of shortage are indicated by circled numbers in Fig. 2.1. Also, operating priorities for periods of water surplus are indicated by numbers in squares. This simplified example helps to understand the concept. However, simulation of large river basin involves a number of details which somewhat complicate the situation. This is briefly reviewed in the next section.

2.3 Interreservoir and Cumulative Priorities

One thing which has to be clearly pointed out is that the physical capacities of the components being modelled overrule the penalty values in determining the actual flow distribution. In particular, diversion structures, reservoir outlet structures or hydroplants have upper flow limits which must be incorporated into the model. As a consequence of this, although the model may try to maintain all reservoirs in the same zone, the channel constraints (physical, operational or both) can create situations where different groups of reservoirs are operating in different zones. This has resulted in classifying reservoirs which operate in the same zones as "reservoirs in the same operational policy". All reservoirs within one operational policy are ranked by having their appropriate penalties made slightly different (usually by 1% to 2%). There was a number of reasons for this:

1) In most cases it is desirable to rank reservoirs in the

same operational policy. This will result in deviations occurring first in the reservoir with the lowest priority, and then progressively increasing to the highest priority reservoir.

2) An attempt to make all the reservoirs in the same operating zone conform to the "equal function relationship" has failed. By assigning identical penalties to the respective zones of these reservoirs, which was to simulate equally distributed violation, the model would not produce a unique answer, although the total system penalty was minimized. The existence of more than one optimal solution is greatly increased if there are many components with identical penalties.

Ranking of the components within the same operating policy is extended to hydropower plants, irrigation components and major withdrawals in a similar manner as in the case of reservoirs. The justification for this is the same as before, i.e. the components are not of the same importance (e.g. two irrigation fields might differ in type or quality of crops). Also, an overall objective is to arrive at a unique solution for a particular run.

2.4 General Model Structure

The WRMM consists of three separate computer programs written in FORTRAN 77 and linked together by shared common data files. These programs are:

1) Data Preparation Program (DPP)
2) Model Simulation Program (MSP)
3) Output Display Program (ODP)

The DPP program is used in the case of simulation runs of more than one year. Its purpose is to access the hydrometeorlogic basic data file of Alberta Environment and create an input data file for every year of MSP simulation run. The ODP program is optional and its purpose is to give some of the outputs in graphical form, which is very useful in case of long simulation runs.

The main part of WRMM is the MSP program whish can be run separately. This program has the following functions:

1) Transformation of model data (physical network, penalty values,inflow and demand) into a capacitated transportation network.
2) Solution of the transformed network by the OKA
3) Checking of the solution flows against the constraints imposed by control structures and hydroplants and reiterating if necessary. Also, conversion of arc flows back into a convenient form of output (elevation for reservoirs, unit of application for irrigation, power for hydroplants and flow for all other components).

Model components and the appropriate physical processes which they simulate are listed in table 2.1.

Natural runoff is simulated as inflows at reservoirs and junctions. Included in the natural runoff is all water contributed to the river under natural conditions.

The nature of reiteration is one of the drawbacks of this modelling concept and it is briefly discussed below:

Outlet structures are often associated with a reservoir or a lake in order to modify the flow below them. There are three types of outlet structures that can be simulated by the model:

- weir flow
- orifice flow
- discharge versus elevation curve

The first two use standard formulas to determine maximum outflow, while the third estimates maximum outflow by interpolation between two closest given points (the whole curve is given as a set of discrete points).

Table 2.1 Component vs modelled processes

Component	*Modelled Processes*
Reservoir	- storage - controlled releases (hydropower excluded) - net evaporation
Hydropower	- power flows - energy generation
Natural Channel	- Channel flow
Apportionment Channel	- Channel flow which is governed by an apport. agreement
Diversion Channel	- flow
Irrigation	- consumptive use - return flow
Junctions	- joining or splitting of flow of two or more components
Major Withdrawal	- consumptive use - return flow
Minor Withdrawal	- consumptive use

Whichever way is used, the actual average elevation per time step must be known in order to calculate the maximum outflow. However, only elevation at the beginning of the time interval is actually known; the elevation at the end of the time interval is yet to be derived by the model. This problem is tackled in the following way: the outflow capacity is first calculated using the elevation at the beginning of the time step. At the end of the time step the new elevation will be derived as a result of balancing inflows, outflow and net evaporation at the reservoir site. The new elevation might be lower than the previous, which indicates that the outflow estimated at the beginning of the time interval was greater than the actual. In this case the starting and the ending elevations are averaged and then compared to the starting elevation. If the difference is over five per cent, the whole optimization for that time interval is repeated, having the capacity of the control structure previously reset.

A similar problem is associated with modelling of hydropower plants, through which the flow is physically constrained by the operating characteristics of the turbine. The maximum flow through the plant is calculated as follows:

For $H<H_r$ $\quad Q_{max}=Q_r\sqrt{H/Hr}$ ....(2.1)

$H>H_r$ $\quad Q_{max}=Q_rH/H_r$ ....(2.2)

where:

H......expected operating head for a time step

H_r......rated head of the plant

Q_r......rated flow of the plant

At the end of the time step the maximum discharge capacity (calculated using the average operating head for that time interval) is compared to the discharge capacity defined by the initial operating head. If the difference exceeds five percent the whole iteration for that time interval is repeated using new maximum discharge capacity calculated with the average operating head.

Both control structure and hydroplant have a potential to create numerous reiterations within one time interval, which can be disastrous in terms of execution efficiency. In order to prevent that, the number of iterations within one time interval is limited to five. It rarely happens that after four iterations the deviation is still over five percent, but if that is the case the warning message is printed out.

2.5 Input Data

There are four basic data sets required for a simulation run. These are:

a) definition of the physical system

b) definition of priorities (penalty system)

c) water supply data

d) water demand data

The water supply data consists of initial reservoir storages, reservoir net evaporation (evaporation minus precipitation) and natural channel flows. The water demand data includes municipal, industrial, hydropower and irrigation requirements.

Data described under a) and b) is contained in the SCF (simulation control file), while water supply and demand data are usually stored in PDF (prepared data file), although for short simulation runs (typically one year), all the necessary data can be conveniently stored in SCF. In case of long simulation runs, the PDF file is created as a result of one DPP (data preparation program) run. This program has the following functions:

1) To access the appropriate (monthly or weekly) hydrometeologic data base file (HBDF) and read the records for simulated years.

    HBDF files consist of natural flows for all of the hydrologic stations in the South Saskatchewan River Basin, starting from year 1912. Included in HBDF are also water demand files, which represent mean monthly (or weekly) flows required for various consumptive uses. These flows were estimated on basis of previous demands, and they represent ideal situation when these demands are met 100%.

    There are two types of data contained in HBDF files; these are the flow values expressed in CMS, and evaporation and precipitation data given in MM. Both monthly and weekly HBDF are organized in the following way: the first line of the record containing the station name, starting year of existing records, number of years of record and finally, the units (CMS or MM). The next four lines are reserved for additional information and constitute the title. After this, records are listed starting with the respective year in the most left column and followed by appropriate data in subsequent columns. Not all the records from HBDF are read by DPP. For example, if the simulation period starts in 1955 and ends in 1970, only records for these 15 years will be read from the appropriate files and used for creation of PDF.

2) Finally, the ultimate goal of DPP is to calculate net inflow for all nodes in the network which are designated as "inflow nodes", whereby net inflow per one time interval is defined as a change of flow with respect to its value contained in the previous time interval. Therefore, net inflows can be both positive or negative.

Specification of natural inflows is of particular importance to this modelling concept. Since all water movements are averaged per time interval and the optimal solution derived by the model is based on this steady state assumption, the only way to represent movements of the water through the system is to monitor changes of natural inflows at certain locations and supply them to the model for each time interval. Locations at which inflows are specified have to be defined as nodes in the WRMM network. In the actual physical system, these locations represent junctions and reservoirs.

Apart from defining net inflow per node with respect to the previous value, there is one more important aspect related to this parameter. The area between two inflow nodes (one being upstream of the other) is the only part of the watershed which contributes to the inflow into the downstream node. Therefore, to assess the net inflow into the downstream node, the natural flows at this node must be subtracted from the sum of natural flows at the upstream node. This technique breaks the whole watershed into a number of small, interconnected watersheds and the natural runoff is simulated as inflow into the most downstream point for each of these elementary watersheds.

2.6 Flow Network Organization

All model components are specified by the user as the external network. In order to apply the OKA solution procedure the external network will be transformed into a directed, circulatory and capacitated network. This is termed the internal network and all data from the external network will be converted into units of flow in the internal network.

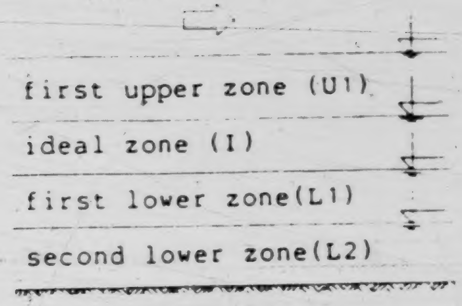


fig 2.2 Flow zones

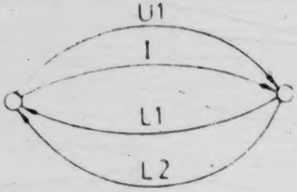


fig 2.3 Flow arcs

The basic idea of this transformation is to split the flow in a physical component (natural channel, for example) into algebraic sum of flows in the "ideal zone" and a finite number of zones above and below the ideal.

Fig 2.3 indicates how a natural channel is represented in the internal network. All zones are specified arbitrarily by the user. The ideal zone implies the amount of flow sufficient to meet all demands (flood excluded). The natural channel flow is thus converted in the internal network as follows:

$$Q = U1+I-L1-L2 \qquad \text{..........................(2.3)}$$

where if the elevation is above the ideal zone, the flows in lower arcs are zero and vice versa. For example, if the elevation is in zone L2, then flow in the ideal arc will be equal to its lower bound, the flow in arc L1 will be equal to its upper bound, the flow in arc L2 will be between the bounds and the flow in arc U1 will equal zero. Therefore, there will always be flow in the ideal arc, which will be either subtracted from flows in reverse arcs, or summed with the flows in the forward arcs (in this example there is only one of them).

The rules of transformation from external to internal network can be briefly summarized as follows:

1) All water movements will be represented by arc flows in the internal network
2) Nodes in the internal network will represent:
    - junctions in the external network
    - locations at which water leaves the system (i.e. evaporation from reservoirs, evapotranspiration from irrigation fields, withdrawals, etc.) in the external network
    - locations at which water is supplied to the system in the external network

The following is an example of a modelled system with one reservoir, one irrigation field, two rivers, one apportionment channel and one major and minor withdrawal. Inflows into the system are specified at two locations at the upstream end of the modelled region. This system is presented in fig 2.4 and its appropriate internal network is shown in fig 2.5.

It can be noted that two more nodes are added in the internal network; these are the system supply node (SSN), from which inflows into the system are specified, and the system balance node (SBN), where arcs leading to this node represent movement of the water outside the system for all components except for reservoirs. For example, the flow in the arcs which link irrigation field with the system balance node represent the volume of water consumed by the irrigation field plus all the losses due to seepage and evaporation from the diversion channel per time interval, which also has to be treated as a flow directed outside the system.

Inflows into the system are specified at only two nodes in this case, both of them being on the boundary of the modelled region. The ideal situation is to choose such locations for these nodes which correspond to locations of the existing hydrologic stations. The actual natural flow data will then represent inflow into the system, which are (in this case) positive. Inflows are generally specified at a number of points in the system. In this simplified example, inflows could also be specified at the junction downstream of the reservoir (node 3 on fig 2.4). If natural flows at this node for one time interval happen to be smaller than the sum of all upstream flows (in this case inflows into nodes 1 and 5), then it means that the sub-watershed bounded by nodes 1,5 and 3 is not contributing to the

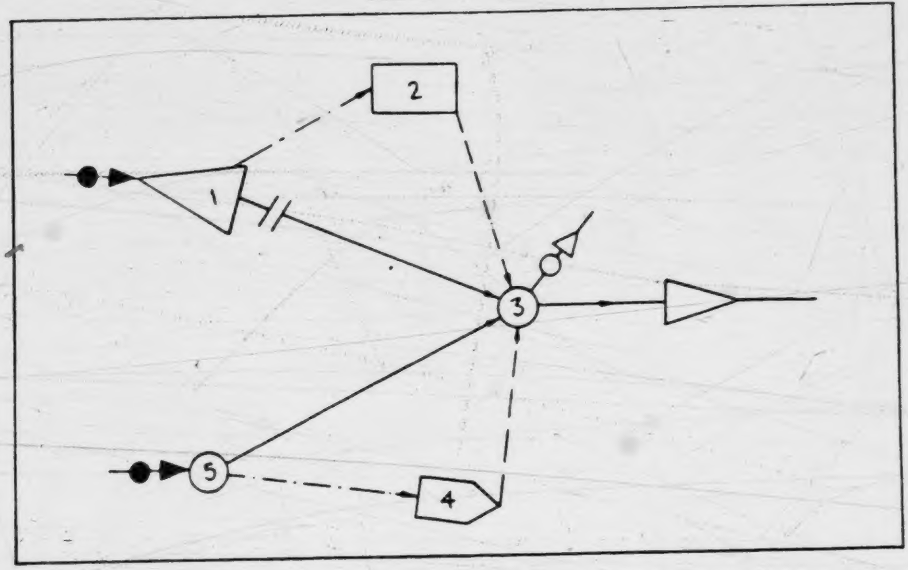
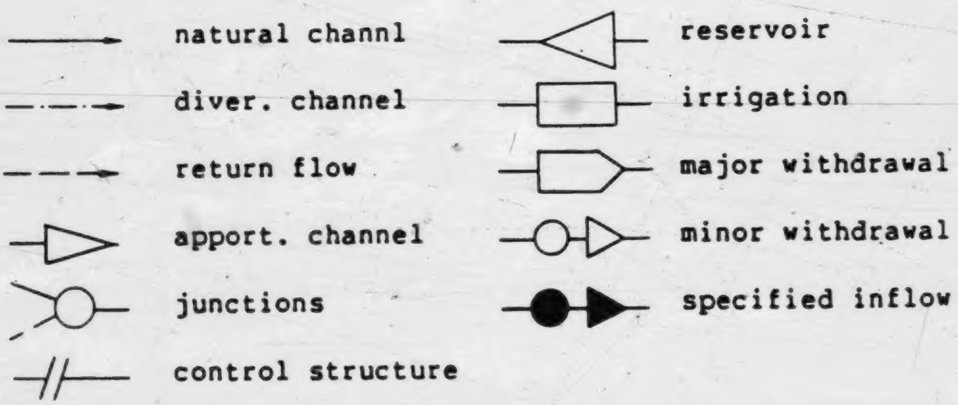


fig 2.4 Example external network

LEGEND


natural channl
diver. channel
return flow
apport. channel
junctions
control structure
reservoir
irrigation
major withdrawal
minor withdrawal
specified inflow

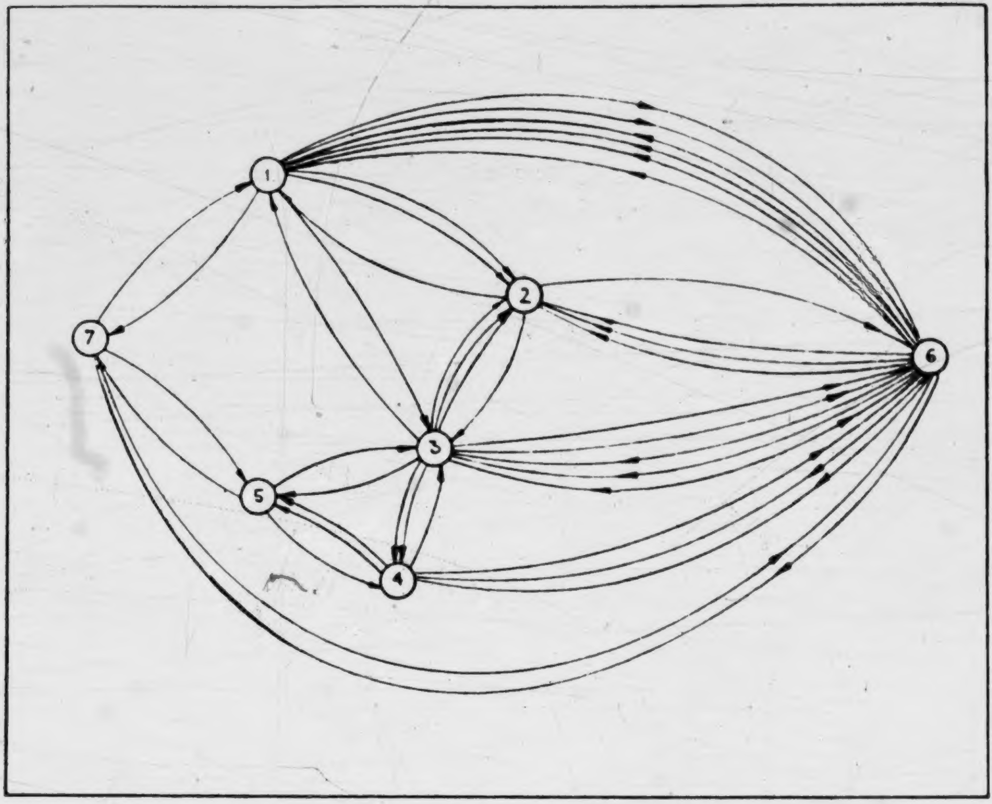


fig 2.5 Example internal network

overall runoff. Consequently, the inflow into node 3 would be assigned negative value (equivalent to estimated losses in the reach between nodes 1,5 and 3).

This explains why there are two inflow arcs with opposite directions. If the current inflow into a node has positive value, it will appear as the flow in the arc directed from system supply node to the actual inflow node, while the flow in the other arc will be set to zero. Conversely, negative inflows will appear as positive flows in arcs directed from the inflow node to system balance node, while the other arc will have its flow set to zero. The importance of organizing internal network in this way is dictated by the requirement that flows be non-negative, imposed by the nature of linear programming. One of the most interesting features of this modelling concept is related to converting change of reservoir storage into flows in reservoir arcs. This topic is further referred to with more details in the numerical example in the next section. For now, it should be noted that reservoir arcs include the following:

- The initial storage arc directed from SBN to the reservoir node. Flow in this arc is set to the flow obtained by dividing the storage volume at the beginning of the time interval by the length of the time interval. Penalty in this arc is set to zero.
- The rule curve arc and the upper zone arcs are directed from the reservoir node to the SBN. The rule curve arc has a penalty of zero while upper zone arcs have positive penalties assigned to each zone.
- The lower zone arcs are directed from SBN to the reservoir node.

The algebraic sum of all these flows can be directed either from SBN to the reservoir node, which represents decrease of storage per time interval, or else if the "summed flow" orientation is from the reservoir node to SBN, the reservoir level is increased per time interval. Reservoirs are the only model components which can have flow directed from SBN towards the system. All other components will have their flows directed towards SBN.

Finally, two arcs with opposite directions link SBN and SSN. One of them always has flow set to zero, while the flow in the other is set to the difference between the total system input and the total volume of water that left the system for a time interval. Normally, it is the arc directed from SBN to SSN which has the flow, while the other arc has flow set to zero. The opposite situation is very unlikely, but still theoretically possible. One could visualize such a desert like river basin in which upstream natural inflows combined with reservoir releases are all lost due to seepage and evaporation. Also, modelling of such basin must not include specification of any requirements that have to be met, since if that is the case the model will declare the problem infeasible. In other words, the model will not be able to allocate the demanded amounts due to the non-availability of water under such extremely dry conditions. These two arcs are necessary in order to satisfy the flow continuity for the system supply and the system balance nodes. To summarize, the internal WRMM network has the following properties:

1) For each node in the network, there is at least one in-coming and at least one out-going arc.
2) All arcs are bounded from below and above. Bounds are constant for each time interval; also, infinitely large upper bounds are set to a finite value of 10,000 CMS.

3) Each arc is assigned a penalty, further referred to as a cost incurred per unit of flow.

Problem of optimal allocation in a river basin is thus converted into solving a minimum cost flow problem for the transformed network. WRMM contains a nested optimization subprogram which solves the problem of routing the inflows through the network in such a way that the sum of penalties times flows for all arcs in the network is minimized. Once the solution is achieved, the resulting flows through all hydroplants and flow control structures are checked against their maximum capacity and the whole process might be repeated as explained earlier. The solution for one time interval represents the starting point for the next time interval, and so forth. The final solution will define the best allocation policy for each time interval (weeks or months). In other words, the WRMM derives a set of flows which deviate from the ideal conditions as little as possible. If deviations have to occur since it is impossible to meet all the demands at a particular time interval, they will first take place at the segments of the system with the lowest prescribed priorities.

2.7 Numerical Example

A one year simulation run with monthly time steps is executed for a fictitious system depicted in fig 2.4. All the necessary information needed to describe this system is given is SCF file enclosed in appendix 3. The SCF file consists of six blocks of data linked in the following order:

$IDENT - system identification sub-file
&SIMCON - simulation control sub-file
$PHYS - physical system sub-file
$PENS - penalty system sub-file
$WATD - water demand sub-file
$WATS - water supply sub-file

The first two sub-files contain basic information about the model, starting date and duration of simulation, the length of time intervals, etc. Physical system data contain the connectivity and component types, while the penalty subfile contains the sizes of all zones and their respective penalties.

The components modelled in this system include:

- reservoir
- flow control structure immediately downstream of the reservoir
- two natural channels
- minor withdrawal (represents demand that has to be met, e.g. municipal supply)
- irrigation component

Representation of the last two components includes diversion and return channels. This flow takes place along arcs 18, 19, 20 and 21 for the irrigation component, arcs 35, 36 and 37 for the industrial component, and arcs 31 for the municipal use component.

The input data file in appendix 3 is followed by two example outputs and an output from subroutine "ARCCHK", which is one of the optional outputs. This subroutine prints upper bound, flow, lower bound and penalty for each arc in the network.

To demonstrate the effect of the penalty values, two runs were executed with the change in reservoir penalties. reservoir initial elevation is set at the rule curve. In the first run, penalties in the zones below the rule curve were set to 10, 20 and 100, respectively. However, in the second run they were increased to 500, 800 and 1000, which gave the reservoir highest priority (excluding the municipal use component). The result of this is visible in the final output. In the first simulation the reservoir level is steadily decreasing, starting from 1030.0 m on January 1,87. In the second run, however, the reservoir elevation remains constant throughout the simulation period, at the expense of other users. For example, the industrial component (reference number 81) has flow greater than zero only in the month of August. Other than that, one should observe that letters A, B, C or D attached to the numbers in the final table indicate in which zone a particular component operates at the given time interval. Zones A, B, C and D represent first, second, third and fourth deviating zone below the ideal, respectively. Conversely, letters X, Y, Z and W indicate operation in zones above the ideal. Also, if a letter is followed by the apostrophe in the final output, this indicates that the corresponding component is operating at the very bottom of the respective zone.

The reservoir penalties and storage-elevation curve are given in Fig 2.6 and Fig 2.7. The rule curve is kept constant, which is not normally the case, but it was done here for simplicity. The first and the second zone below the rule curve change their sizes with time. In the first simulation run, the reservoir level decreases through all zones, and the letters are printed accordingly.

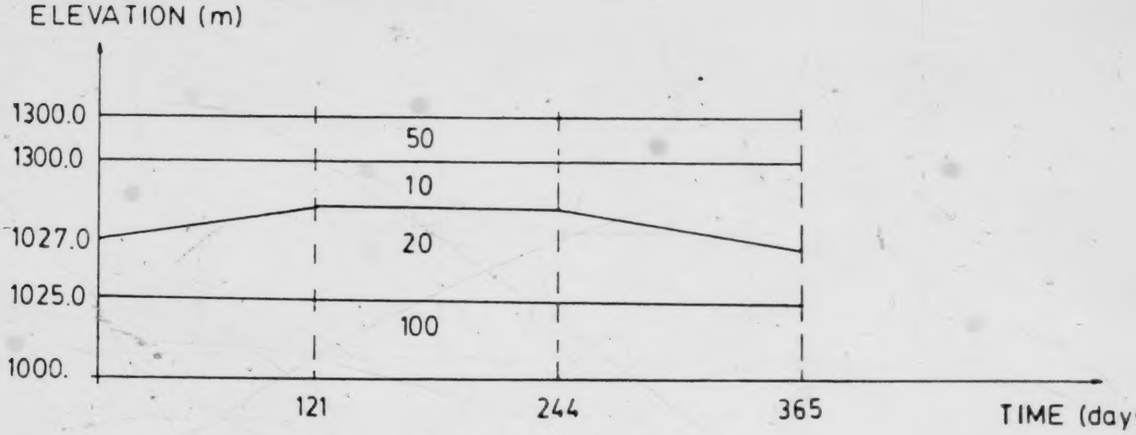


fig 2.6 Example reservoir zones

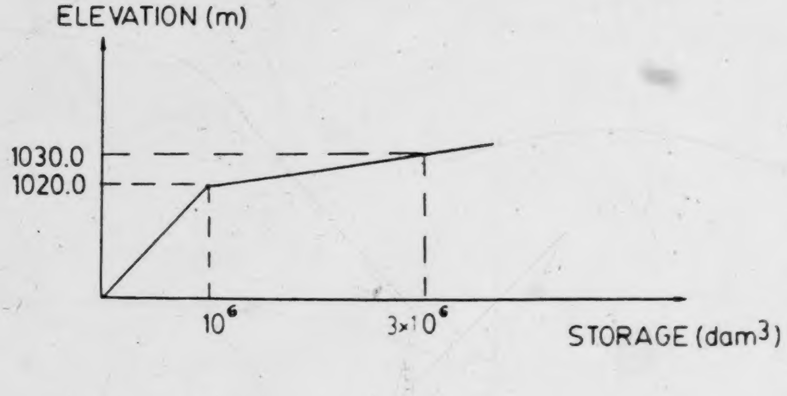


fig 2.7 Example elevation vs storage curve

The starting elevation at the beginning of the first time step is set at the rule curve level. Dividing the rule curve storage with 31 days for month of January gives a "storage equivalenced" flow of 112.007 CMS, which is the flow in arcs 7 and 8 (the rule curve arc and the initial storage arc). The outflow from reservoir is represented as a flow from the system balance node towards the reservoir node, and the average outflow for month of January is 3.58 CMS (arc number 9). Each unit of this flow is penalized with the penalty of 10, and the new elevation at the end of the first time interval is 1029.52 m. The storage which corresponds to this elevation is divided with the length of the next time interval (month of February) and it represents the flow in the initial storage arc for the second time interval. The rule curve arc is set in the similar manner. Reservoir storage is further reduced in the second time interval for the average flow of 2.26 CMS. However, the penalized flow in arc 9 is 6.224 CMS, which represents storage equivalenced flow of accumulated deviation for all previous time steps.

Finally, the basic WRMM output can be used as input for other studies, since it is easy to distinguish deficits for all components in each time interval. Different deficits are subject to economic analysis which could assess the viability of constructing a new component with respect to its size or location.

2.8 Final Remarks

A number of critical observations regarding WRMM is presented in this section which should be helpful to those who are interested in the further development of this modelling concept. A number of difficulties associated with this modelling concept were already mentioned before, such as the steady state data representation and reiterations on control structures and hydropower plants. Additional remarks are reviewed in the following.

2.8.1 Natural Channel Flows

As explained earlier, natural channels are represented in the model as a number of sequentially linked components, joined by nodes. Some of these nodes may be specified as "inflow nodes", and their respective inflows for each time step are assigned to arcs which link the system supply node and the individual inflow nodes. The result of this is that the natural flow between the two adjacent nodes is treated as constant along the reach, i.e. the increase in flow due to runoff can only be specified in increments, which in reality this increase is continual. Observe for example a natural channel downstream of the reservoir in fig 2.4. It could represent a reach of 100 km or more in reality. If at the particular time interval no water is released from the reservoir, the final output will indicate a flow of zero for this component.

However, the flow of zero will exist only immediately downstream of the reservoir. After 100 km, natural runoff can build up a considerable flow, but this will only appear as inflow into the downstream node (coming through inflow arcs) and it will affect the downstream component, in this case the apportionment channel. Therefore, the final output is not completely accurate in the case of natural channels. This problem cannot be avoided, but it can be reduced using river reaches of smaller length. This, however, reduces the computational efficiency since the size of the network is increased.

2.8.2 Infeasible Solutions

Many users are discouraged by this modelling concept after they face this problem, and a comprehensive way to deal with it is yet to be developed.

It is essential to realize that, although all arc bounds are implicitly set by the user in the input data file, it is still impossible to know whether a feasible solution exists for all time intervals. Resetting of the arc bounds is done by the model on basis of the previous time step solution and inflows and demands for the current time step. In a situation when the natural supply is lower than the demand which has to be met (described by sum of the minor withdrawal components), the algorithm will not be able to solve the problem, and it will declare infeasibility. This prompts the user to examine the input data file and determine the reason for infeasibility. The current procedure is not too helpful and only experienced users can resolve this problem, most of the time using the "trial and error" procedure. A separate subroutine which will analyze the capacitated network in the case of infeasibility, transform the results of this analysis into the external network and offer alternate courses of action to resolve this problem is currently being developed.

2.8.3 FORTRAN 77 vs Other High Level Programming Languages

WRMM is written in FORTRAN 77. This has its limitations which negatively affect the whole model. These limitations are associated with inability to use more appropriate data structure, which could improve the following:

1) Operating memory allocation could be easily made adjustable, which could contribute to more efficient utilization of the model. The current version installed on the Terrace Computing Center Mainframe allocates 2.5 MB of memory for each run, even though only 5 % of it may be actually needed for small size networks. This consideration is of particular importance to those who intend to use WRMM on mini or microcomputers.

    However, this is not the major concern since it is possible to override this FORTRAN limitation using a number of different techniques. A considerable advantage of languages such as PASCAL or C language would be in ability to use record data structures, which would eliminate arrays used in FORTRAN and consequently reduce the number of referencing and array operations in the program, thus improving its overall efficiency. Detailed discussion of these advantages requires understanding of how the current FORTRAN data structure works; it is therefore placed at the end of this study.

3. THE OUT-OF-KILTER ALGORITHM OF FORD AND FULKERSON

3.1 Introduction to Linear Programming

This section reviews basic definitions of linear programming, duality and the theorem of complementary slackness, which are the theoretical foundations of the Out-of-Kilter Algorithm of Ford and Fulkerson.

3.1.1 Definition of a Linear Program

The theory of linear programming is concerned with the solution of linear inequality systems. Its rapid development began in the late 1940s and it has been substantially enhanced by the development of electronic computers.

A linear program can be defined as a decision problem of finding an extreme value of a linear form, subject to linear inequality constraints. The following mathematical expressions represent an example of this statement (in matrix notation):

$$\text{minimize} \quad \mathbf{c} \cdot \mathbf{x} \quad \ldots\ldots (3.1)$$

$$\text{subject to} \quad \mathbf{a} \cdot \mathbf{x} \geq \mathbf{b} \quad \ldots\ldots (3.2)$$

$$\mathbf{x} \geq 0 \quad \ldots\ldots (3.3)$$

where (3.1) is called the objective function while (3.2) and (3.3) are called constraints of the system. Matrices in these expressions are of the following type:

- c is a row matrix of size n
- **x** is a column matrix of size n
- a is a matrix with m rows and n columns

- **b** is a column matrix of size m

The notation used here follows mainly the lines of Jarvis and Bazaraa [43]. Matrices **c**, **a** and **b** are given while matrix **x** is unknown. Expressions (3.1), (3.2) and (3.3) define a linear program in its *canonical* form. There are some important properties of linear programs which will be briefly outlined here:

a) Expression (3.2) can be converted into a matrix equation by defining a set of slack variables **s**, such that:

$$\mathbf{a}\cdot\mathbf{x}-\mathbf{s}=\mathbf{b}$$

If this is the case, the linear program defined by (3.1), (3.2) and (3.3) is in its *standard* form and the system must be solved for both **x** and **s**.

b) If matrix **c** is multiplied by -1, the problem of minimization becomes the problem of maximization.Also, if expression (3.2) is multiplied by -1, the inequality changes from "greater or equal" to "less or equal".

c) Finally, expression (3.3) indicates that all the variables have to be nonnegative. However, some of the variables in actual optimization problems may be unrestricted in sign. In such cases the following transformation can be applied:

$$x = x' - x'' \quad \text{......................} (3.4)$$

where both x' and x" are nonnegative. Any value of x can be presented as a combination of two nonnegative values of x' and x".

To summarize, any linear program can be brought into a form defined by (3.1), (3.2) and (3.3) regardless of the combination of equalities and inequalities in the constraints on the variables or restrictions in their sign. Any matrix **x** which satisfies (3.2) and (3.3) is called a feasible solution. The problem for linear programming is to isolate a feasible solution which also satisfies equation (3.1); this is referred to as an optimal solution.

3.1.2 The Concept of Duality

A fundamental property of a linear program is that another linear program, called its "dual", can be associated with it. John Von Neumann [44], the founder of the theory of games, is credited with having first postulated the existence of a dual linear program. Developments in the theory of duality have led to new and more efficient algorithms like the Dual-Simplex method, the Primal-Dual algorithm and the Out-of-Kilter algorithm.

The dual program of the linear program defined by (3.1), (3.2) and (3.3) is defined by:

$$\text{maximize} \quad \mathbf{w}\cdot\mathbf{b} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.5)$$

$$\text{subject to} \quad \mathbf{w}\cdot\mathbf{a} \leq \mathbf{c} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.6)$$

$$\mathbf{w} \geq 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.7)$$

where **w** is the dual variable and **a**, **b** and **c** are the matrices previously defined in (3.1) and (3.2).

The dual program defined by (3.5), (3.6) and (3.7) is actually another form of the primal program defined by

(3.1), (3.2) and (3.3). To show this, the following transformations are applied:

1) Multiply expressions (3.5) and (3.6) by -1 and transpose both sides of expressions (3.5), (3.6) and (3.7); the transpose matrix is denoted by the superscript *t*. The dual program then becomes:

$$\text{minimize} \quad (-b^t)\cdot w^t \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.5')$$

$$\text{subject to } (-a^t)\cdot w^t \geq (-c^t) \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.6')$$

$$w^t \geq 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.7')$$

2) If (3.5'), (3.6') and (3.7') is considered a primal program, its dual can be defined according to the relation between (3.1), (3.2), (3.3) and (3.5), (3.6) and (3.7) as:

$$\text{maximize} \quad x^t\cdot(-c^t) \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.5'')$$

$$\text{subject to } x^t\cdot(-a^t) \leq (-b^t) \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.6'')$$

$$x^t \geq 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.7'')$$

3) Finally, if both sides of expressions (3.5"), (3.6") and (3.7") are multiplied by -1 and then transposed, this program is converted into the primal program (3.1), (3.2) and (3.3). This proves the theorem that " dual of the dual is the primal", i.e. both the dual and the primal are different forms of the same program.

Moreover, if solutions of the primal and its dual program are denoted by x_{opt} and w_{opt} respectively, then:

$$c\cdot x_{opt} = w_{opt}\cdot b \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.8)$$

To show this, assume that x_f and w_f are feasible solutions to the primal and dual programs respectively. Then multiply expression (3.2) on the left by $\mathbf{w}_f$ and expression (3.6) on the right by $\mathbf{x}_f$, to give:

$\mathbf{cx}_f \geq \mathbf{w}_f\mathbf{ax}_f \geq \mathbf{w}_f\mathbf{b}$ ....(3.8.1)

This relationship shows that the value of the objective function for any feasible solution to the primal minimization problem is always greater or equal than the value of the objective function for any feasible solution to the dual maximization problem. The "greater or equal" sign in this expression may or may not become "equal" only when $\mathbf{x}_f$ and $\mathbf{w}_f$ respectively minimize the objective function of the minimization problem and maximize the objective function of the maximization problem. Then $\mathbf{x}_f$ and $\mathbf{w}_f$ would also be optimal, and the foregoing inequality would become:

$\mathbf{cx}_f = \mathbf{w}_f\mathbf{ax}_f = \mathbf{w}_f\mathbf{b}$ ....(3.8.2)

For optimal $\mathbf{x}_f$ and $\mathbf{w}_f$ this equality is always valid, which follows from Khun-Tucker optimality conditions [45], [46].

Expression (3.8.2) is broken down into two equations:

$\mathbf{w}(\mathbf{ax}-\mathbf{b}) = 0$ ....(3.9)

and $(\mathbf{c}-\mathbf{wa})\mathbf{x} = 0$ ....(3.10)

where $\mathbf{w}=\mathbf{w}_{opt}$ and $\mathbf{x}=\mathbf{x}_{opt}$; the only way in which these two equations will hold is if at least one of the factors in the dot product on the left hand side of the equations (3.9) and (3.10) is equal to zero for each row *i* of the matrix equations (3.9) and (3.10). This requires that:

$w_i > 0 \rightarrow [a_{i,j}][x_j] = b_i, \quad j=1,n$ ....(3.9.1)

$w_i = 0 \rightarrow [a_{i,j}][x_j] > b_i, \quad j=1,n$ ....(3.9.2)

$x_j > 0 \rightarrow [w_i][a_{i,j}] = c_j, \quad i=1,m$ ....(3.10.1)

$$x_j = 0 \rightarrow [w_i][a_{i,j}] < c_j, \quad i=1,m \ldots\ldots\ldots\ldots (3.10.2)$$

From the above conditions the weak theorem of complementary slackness [47], [48] can be stated as: if the variable in one problem is positive, then the corresponding constraint in the other problem must be "tight" (i.e. it must be equality); conversely, if the constraint in one problem is not tight, then the corresponding variable in the other problem must be zero.

This gave rise to algorithms that attempt to bring both primal and dual variables into one of the foregoing conditions, in which case an optimal solution is found. The advantage of such algorithms is in reducing the number of steps (iterations) before an optimal solution is reached. At first sight this might seem to be a tradeoff in the wrong direction, since both the primal and dual problems have to be solved, when usually only one of them is of interest. However, these algorithms offer the option to tackle the same problem from two different perspectives (primal and dual), and use whichever is more convenient at any particular point. This is analogous to having two routes to get from point A to point B in each iteration, and choosing the shortest. Classical linear programming algorithms solve only the primal program, which is analogous to having only one route from A to B.

3.2 Introduction to Network Minimum Cost Flow Problem

The Out-of-Kilter algorithm (OKA) is an efficient linear programming routine for solving minimal cost flow problems in networks. This routine can only be applied to networks with certain properties, which will be briefly outlined here.

A network, or linear graph G=[N,A] is a finite set of N elements together with subsets A'⊂ A associated with each element of set N. The elements of N are known as: junctions,points, vertices or nodes; the elements of A are called arcs, links, edges or branches. The expressions "nodes" and "arcs" will be used further in this text.

The OKA routine is applied exclusively to directed, circulatory and capacitated networks. The term "directed" indicates that flows along a particular arc are allowed in only one direction. The term "circulatory" means that for each node "i" in the network there is at least one incoming arc and at least one outgoing arc. It is usually only the first and the last node in the network that don't satisfy this condition for most of networks in general. Such non-circulatory networks can be made circulatory simply by adding one arc between the starting and the ending node.

Finally, a "capacitated" network will have "capacities" defined for each arc, which will further be referred to as the "upper bounds" and the "lower bounds". Capacities for each arc are needed to represent real world network flow limitations, which could be imposed by existence of pumps, turbines, or other physical constraints.

Having defined a directed, circulatory and capacitated network, one more variable for each arc is needed to formulate the minimum cost flow problem. This variable represents the fictitious cost of sending a unit of flow from node i to node j along an arc (i,j). In some networks the cost of flow has an economic representation (e.g. the cost of pumping along the pipeline). However, this does not have to be the case and the cost of flow along each arc can be assigned quite arbitrarily. Whatever the case, different costs (or penalties) can be used to define certain preferences for routing the flow through the network.

A set of flows in the network which satisfies continuity of flow for every node is termed *a circulation*. If such a set of flows also lies between the prescribed bounds on every arc, the circulation is said to be feasible. The problem of minimal cost flow in a network is to find such a feasible circulation which also minimizes the total cost of flow in the network, where the total cost is the sum of flow multiplied by cost for all arcs in the network. There might be one or more circulations that will meet all three requirements (prescribed bounds, node conservation and minimal total penalty). These circulations are called optimal, and the OKA algorithm is a technique of reaching an optimal circulation, starting from any circulation for a given network.

3.3 The Out-of-Kilter Algorithm

3.3.1 Theoretical Basis of the Algorithm

The out-of-kilter algorithm [36], [37] is a generalized primal-dual algorithm for network flow problems. The problem which is solved by the OKA can be mathematically defined as follows:

$$\text{Minimize} \quad \sum_{i,j} c_{i,j}x_{i,j} \qquad (i,j) \in A \quad .(3.11)$$

$$\text{Subject to} \quad \sum_{j} (x_{i,j} - x_{j,i}) = 0 \qquad (j) \in 1,2, ..N \quad (3.12)$$

$$x_{i,j} \geq L_{i,j} \qquad (i,j) \in A \quad ..(3.13)$$

$$x_{i,j} \leq K_{i,j} \qquad (i,j) \in A \quad ..(3.14)$$

where:

A - set of all arcs in the network

N - set of arcs associated with a node

$x_{i,j}$ - flow from node i to node j

$c_{i,j}$ - cost per unit of flow from i to j

$L_{i,j}$ - lower bound of flow from i to j

$K_{i,j}$ - upper bound of flow from i to j

This notation allows for the existence of only one arc from node i to node j. There can be more than one such arc in general. However, this notation is maintained for compliance with the referenced literature.

Expression (3.12) represents the conservation of mass equation for each node in the network. Since the right hand side of this equation is always equal to zero the flow does not enter or leave the network at any node, but rather "circulates" inside the network. Therefore, if the network is not circulatory, it will be impossible to satisfy constraint (3.12). Expressions (3.13) and (3.14) impose the limits of flow on each arc. Since flows are nonnegative it is assumed that $K_{i,j} \geq L_{i,j} \geq 0$ for all arcs (i,j). Finally, expression (3.11) is the objective function of the primal program (3.11)-(3.14).

As previously stated, every primal linear program can be associated with its dual program. The rules for conversion from primal to dual and vice versa can be found in the related literature [49], [50] and they will not be discussed here. The dual of the primal program (3.11)-(3.14) is stated as:

$$\text{Maximize} \quad \sum_{i,j} (L_{i,j} u'_{i,j} - K_{i,j} u''_{i,j}) \qquad (i,j) \in A \quad .(3.15)$$

$$\text{Subject to} \quad u_i - u_j + u'_{i,j} - u''_{i,j} \leq c_{i,j} \qquad (i,j) \in A \quad (3.16)$$

$$u_i \ \text{unrestricted in sign,} \qquad i \in N \quad ..(3.17)$$

$$u'_{i,j},\ u''_{i,j} \geq 0 \qquad ..(3.18)$$

Dual variables u_i (known as "node potentials", or "multipliers") are associated with node conservation equations (3.12) and dual variables $u'_{i,j}$ and $u''_{i,j}$ are associated with constraints (3.13) and (3.14), respectively.

Apply the complementary slackness theorem to this problem: a feasible circulation x and dual variables u_i, $u'_{i,j}$, $u''_{i,j}$ satisfying (3.15)-(3.18) constitute an optimal circulation if and only if they satisfy the relations:

$$u'_{i,j} (x_{i,j} - L_{i,j}) = 0 \quad \text{.......................}(3.19)$$

$$u''_{i,j} (K_{i,j} - x_{i,j}) = 0 \quad \text{.......................}(3.20)$$

$$[c_{i,j} - u_i + u_j - u'_{i,j} + u''_{i,j}] x_{i,j} = 0 \quad \text{.......}(3.21)$$

In other words, for every arc (i,j) one of the (mutually exclusive) conditions has to be satisfied:

$$x_{i,j}=L_{i,j} \quad \Rightarrow \quad u''_{i,j} = 0 \quad \text{...............}(3.19.1)$$

$$\text{and} \quad u'_{i,j} = c_{i,j} - u_i + u_j \quad \text{....}(3.19.2)$$

$$L_{i,j}<x_{i,j}<K_{i,j} \quad \Rightarrow u''_{i,j} = u'_{i,j} = c_{i,j} - u_i + u_j = 0 \quad (3.20.1)$$

$$x_{i,j}=K_{i,j} \quad \Rightarrow \quad u'_{i,j} = 0 \quad \text{...............}(3.21.1)$$

$$\text{and} \quad u''_{i,j} = -c_{i,j} + u_i - u_j \quad \text{...}(3.21.2)$$

Finally, introducing the potential function (or marginal cost) as a dual variable function in the form of:

$$\bar{c}_{i,j} = u'_{i,j} - u''_{i,j} = c_{i,j} - u_i + u_j \text{..........}(3.22)$$

the theorem of complementary slackness can be reformulated as follows: A circulation **x** is optimal if and only if it is possible to find a potential (u_i) such that, for every arc (i,j) one of the following conditions is satisfied:

$$\bar{c}_{i,j} > 0 , \quad x_{i,j} = L_{i,j} \quad \text{...............}(3.23)$$

$$\bar{c}_{i,j} = 0 , \quad L_{i,j} \le x_{i,j} \le K_{i,j} \quad \text{..............}(3.24)$$

$$\bar{c}_{i,j} < 0 , \quad x_{i,j} = K_{i,j} \quad \text{...............}(3.25)$$

Arcs which satisfy conditions (3.23)-(3.25) are called conforming, or "in-kilter". The out-of-kilter algorithm is the sequence of changing flows $(x_{i,j})$ and node potentials (u_i) such that each arc is brought into one of the three in-kilter states, without violating the kilter state of arcs previously brought in-kilter. Values of marginal cost $\bar{c}_{i,j}$ and flow which do not satisfy (3.23)-(3.25) will be non-conforming, or "out-of-kilter". Hence the name "out-of-kilter algorithm".

An example is presented in [51], giving an economic interpretation of the complementary slackness theorem given by (3.23)-(3.25). In an oil pipeline network the cost of sending a unit of oil from node i to node j is the maintenance and the pumping cost. Also, the oil can be sold at a price π_i at every node, such that:

$$\pi_i = -u_i \qquad (3.22')$$

Variable π is known as the pricing vector. Marginal cost was originally defined by Ford and Fulkerson using π_i as:

$$\bar{c}_{i,j} = c_{i,j} + \pi_i - \pi_j \qquad (3.22'')$$

The following interpretation of in-kilter conditions (3.23)-(3.25) can be observed:

If a unit of oil (commodity) is retained at node i, it can be sold-incurring a profit of π_i. On the other hand, if a unit of oil is moved to j, it can be sold at node j incurring a profit of π_j minus cost of shipment of $c_{i,j}$. Since the cost of shipment is constant, the only variable which indicates whether it is profitable to send a unit of flow or not is the node price π. In particular, if marginal cost equals zero, then the profit of selling a unit of oil at node i equals the profit of selling the same unit at node j minus the cost of shipment. Therefore, any amount of flow from i to j is equally profitable, as long as it is between prescribed minimum and maximum (i.e. arc bounds). This case is mathematically expressed by (3.24).

The other two cases follow the same reasoning. If the profit incurred by selling a unit of oil at node i is greater than the net profit of selling it at node j, then the flow to j should be kept to its minimum, e.g. equal to the lower bound (expression (3.23)); conversely, if the net profit of selling a unit of oil at node j is greater than the profit of selling it at node i, the flow should be equal to its upper bound (expression (3.25)).

Additional remarks are needed with regard to this example:

1) Selling a unit of oil at the node indicates that oil leaves the network at that node. To represent this in a circulatory network, additional nodes must be added to the network, termed "system supply" and "system balance" node. The nodes at which oil is sold must have an additional arc leading to the system balance node; flow in this arc represents the amount sold at the node,
2) "Profit" in the forgoing example is envisaged as "net profit", i.e. gross profit incurred by sales minus the cost of shipment, which is always contained in the value of the marginal cost.

Therefore, the only way of increasing profit is by reducing the cost of shipment. Otherwise, large increase in values of the node prices π_i, would be a trivial way of increasing profit.

This example was designed as help in understanding the complementary slackness theorem to those who are not familiar with the concept of duality in linear programming. It attempts to provide a tangible meaning to node potentials, whose values should be visualised as a flow regulating factor for each arc associated with a particular node. Node potentials could also be visualized using other parameters. For example, temperature (at the node) might be a flow regulating factor for some networks, where in order to change flow in an arc its node temperature would have to be adjusted first. To summarize, node potentials should simply be understood as variables associated with each node which can be very useful in the search for an optimal solution.

3.3.2 General Description of the Method

The algorithm starts with any circulation **x** and any set of node potentials u_i. Each arc in the network is examined, one by one, for its compliance with the kilter conditions (3.23)-(3.25). Once an out-of-kilter arc is encountered, the algorithm attempts to bring it into one of the in-kilter conditions, without increasing the deviation of other out-of-kilter arcs and without violating the condition that

x is a circulation. In order to modify the flow of an out-of-kilter arc, a convenient path (called a flow augmenting path) must be found within the network, such that a cycle of arcs is formed (including the starting out-of-kilter arc). The flows are then changed on all arcs on the flow augmenting path by the same amount. This results in bringing all out-of-kilter arcs closer to the in-kilter states while maintaining the continuity of flow for all nodes.

The actual search for a flow augmenting path is carried out by alternate use of a labelling procedure and a potential change procedure.

The *labelling procedure* consists of generating a tree of arcs (called a labelling tree) along which flow may be changed without further violation of their kilter status. Suppose that arc (s,t) is out-of-kilter and that its flow needs to be increased in order to bring it in kilter. The desired increase in flow is finite and it depends on the current value of the marginal cost $\bar{c}_{s,t}$ and flow $x_{s,t}$. For example, if the starting flow on arc (s,t) is zero, there are two possibilities:

1) $\bar{c}_{s,t} \geq 0$, in which case the desired increase of flow is equal to $L_{s,t}$
2) $\bar{c}_{s,t} < 0$, and the desired increase of flow equals $K_{s,t}$

Both of these actions would bring the arc (s,t) in kilter. In order to preserve the conservation of flow at nodes s and t, a cycle of arcs containing arc (s,t) must be isolated, along which flows can be changed by the same amount, without violating the kilter status of any arc in the cycle.

The labelling procedure starts with labelling the starting and ending node of arc (s,t) with destination n and origin 0, respectively (fig 3.1). Starting from the origin (now a labelled node), consider all arcs associated with this node, and isolate the following:

1) For all arcs leading out of the labelled node t, examine their flow and marginal cost, and find the ones which belong to the states λ or μ, where:

   λ: $\bar{c}_{i,j} > 0$ and $x_{i,j} < L_{i,j}$

   μ: $\bar{c}_{i,j} \le 0$ and $x_{i,j} < K_{i,j}$

   Arcs that belong to one of these two states will improve their kilter status if their flows are increased. In particular, arcs in state λ will be brought in kilter if their flow is increased by

   $(L_{i,j} - x_{i,j})$.

   Any further increase in flow would violate their kilter status. Similarly, arcs in state μ could have their flows increased by an amount equal to

   $(K_{i,j} - x_{i,j})$

   without further violation of their kilter status. It is important to note that one of the in-kilter states is implicitly contained in the condition μ. If marginal cost equals zero and the flow is between the bounds, such an arc is both in kilter and in state μ. Flow on such an arc can be increased until it equals the upper bound and its kilter status will remain unchanged.

2) For all arcs leading into the labelled node t, examine their flow and marginal cost, and find the ones that belong to the states ν or ρ, where:

   ν: $\bar{c}_{j,i} \geq 0$ and $x_{j,i} > L_{j,i}$

   ρ: $\bar{c}_{j,i} < 0$ and $x_{j,i} > K_{j,i}$

   These arcs will have their kilter status improved if their flows are decreased; in particular, an arc in state ν will allow a maximum decrease of flow equal to:

   $(x_{j,i} - L_{j,i})$

   while the arcs in state ρ will allow a maximum decrease of

   $(x_{j,i} - K_{j,i})$

   Again, the in-kilter state (3.24) will be implicitly contained in state ν if marginal cost equals zero and flow is between the bounds.

Conditions λ, μ, ν, ρ are called *label eligible conditions*, (or states). Consequently, arcs that belong to one of these states are called *label eligible arcs*.

Having identified the label eligible arcs associated with the starting node (the head node of the arc being currently brought in-kilter), the labelling procedure proceeds as follows: assign a label to the head node of each arc that belongs to sets λ or μ, and to the tail node of each arc that belongs to sets ν or ρ, such that the label consists of three pieces of information:

$(m, \pm, \alpha_i)$

where m is an arc number, plus or minus refers to the direction of an arc and α denotes the maximum permissible flow change associated with a label eligible arc.

The next step in the labelling procedure is to search for label eligible arcs starting from nodes that have been previously labelled. For example, in Fig 3.1 there are two label eligible arcs associated with the origin, and two nodes (3 and 7) were labelled in the first cycle. The search for label eligible arcs now continues from these two nodes; it is found that there are two label eligible arcs associated with node 3, and three label eligible arcs associated with node 7. However, only four new nodes are labelled, since arcs 11 and 12 lead to the same node. Once node 9 is labelled by arc 11, there is no need to label the same node again by arc 12. The final goal of the labelling procedure is to label the destination (tail node of the starting arc). This situation is called *breakthrough* and it is shown in Fig 3.2.

Starting from destination n, it is possible to retrace the flow augmenting path back to origin 0, since each node is labelled with an arc number. In doing so, the amounts of permissible flow change for all arcs on this path are compared in order to find the minimum α_i, where i is the arc number of an arc that lies in the flow augmenting path.

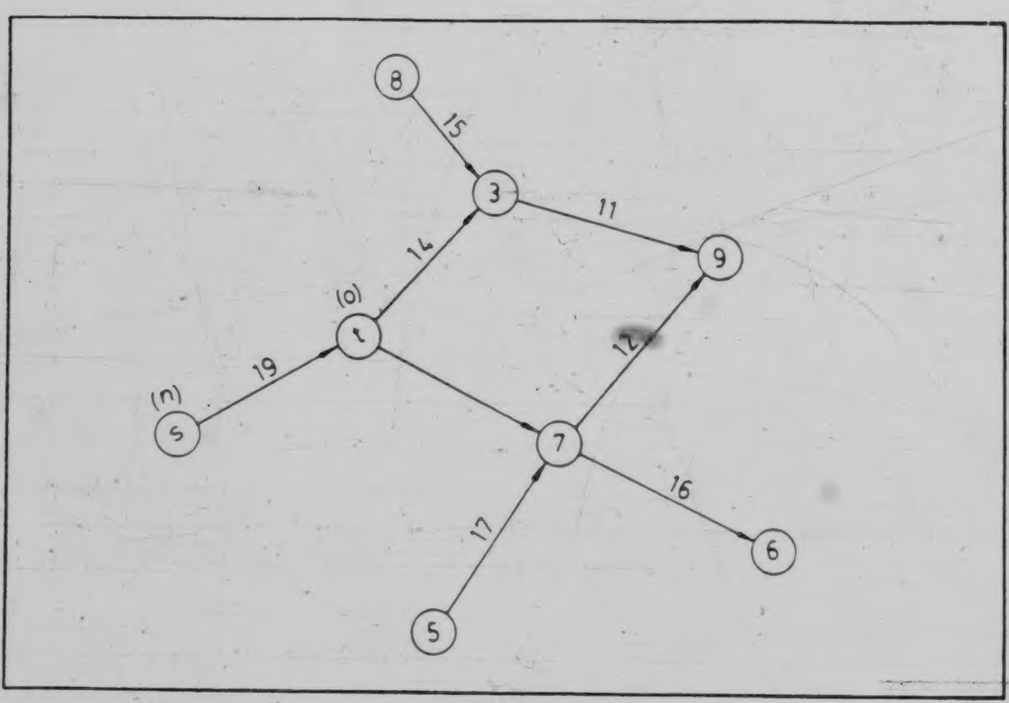

8
15
3
11
9
14
(0)
t
12
19
(n)
s
7
16
17
6
5

fig 3.1 Labelling procedure - non-breakthrough

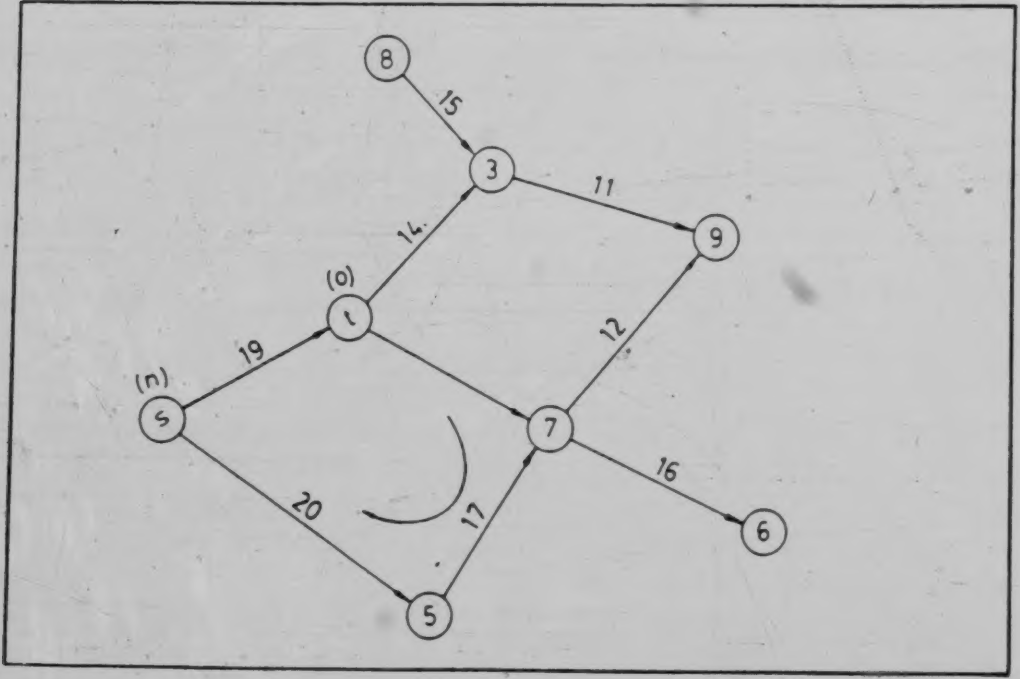

8
15
3
11
14.
9
(0)
t
12
19
(n)
s
7
16
20
17
6
5

fig 3.2 Labelling procedure - breakthrough

Having identified the minimum, flow on all arcs along the flow augmenting path is changed by an amount equal to the detected minimum. In doing so, if the arc on which this minimum was detected was out-of-kilter before the flow change, it will be in-kilter after the flow change. There is also a possibility that this arc was in-kilter (either state μ or ν with marginal cost of zero) before the flow change, in which case it will still remain in kilter.

To summarize, each breakthrough will bring all out-of-kilter arcs on the flow augmenting path somewhat closer to kilter, but none of them necessarily in-kilter. This type of flow change has two important properties:

1) Flow continuity will not be violated at any node.
2) The kilter status of each arc will be either improved or it will remain unchanged.

In general, several breakthroughs are performed in order to bring the starting arc (s,t) in-kilter. Also, each labelling procedure will not necessarily result in a breakthrough. For example, starting from the origin in Fig 3.1 nodes 7 and 3 are labelled in the first cycle, and nodes 8, 9, 6 and 5 in the second cycle. If, however, all arcs associated with nodes 8, 9, 6 and 5 are examined in the third cycle and none of them are found label eligible, the following problem is encountered: flow change (without violating kilter status) is possible only on label eligible arcs, yet any attempt to change flow on them (including the arc (s,t)) would violate the node conservation for node s.

The solution is to extend the labelling tree by changing node potentials of the labelled nodes in such a way that at least one label ineligible arc becomes label eligible. This is done by the potential change procedure. The *potential change procedure* consists of changing node potentials of all labelled nodes, such that at least one more arc becomes label eligible without further violation of the kilter status of any arc. This procedure is applied after the labelling procedure has resulted in a *non-breakthrough* (fig 3.1).

The potential change procedure is applied in the following steps:

1) Denote all labelled nodes as set $\bar{I}$, and all unlabeled as set I and isolate all arcs whose one node is labelled and the other unlabelled:

   $\{(i,j) \mid i \in \bar{I},\ j \in I\}$, or:

   $\{(i,j) \mid i \in I,\ j \in \bar{I}\}$

2) Classify all these arcs into sets S and R, where:

   $S = S_1 \cup S_2$ and $R = R_1 \cup R_2$, such that:

   $S_1 = \{(i,j) \mid i \in \bar{I},\ j \in I\ \bar{c}_{i,j} > 0,\ x_{i,j} \le K_{i,j}\}$ ..(3.23)

   $R_1 = \{(i,j) \mid i \in \bar{I},\ j \in I\} - S_1$ ..(3.24)

   $S_2 = \{(i,j) \mid i \in I,\ j \in \bar{I}\ \bar{c}_{i,j} < 0,\ x_{i,j} \ge L_{i,j}\}$ ..(3.25)

   $R_2 = \{(i,j) \mid i \in I,\ j \in \bar{I}\} - S_2$ ..(3.26)

   Note that an arc which belongs to set S will become label eligible provided that its marginal cost becomes zero, except for the limit cases when flows are equal to upper or lower bound, respectively.

3) Compare values of marginal cost for all arcs from set S and find the one whose value is the closest to zero. The marginal cost of this arc becomes the value θ by which the node potentials of all labelled nodes are increased.

4) Change the node potentials of the labelled nodes only, by adding θ to each of them. Note (from expression (3.22)) that arcs whose both nodes were labelled will have their marginal cost unchanged, and so will arcs whose both nodes were unlabeled. Therefore, the new marginal costs are:

$$\bar{c}'_{i,j} = \bar{c}_{i,j} - \theta, \quad (i,j)\epsilon S_1 \cup R_1 \quad ....(3.27)$$

$$\bar{c}'_{i,j} = \bar{c}_{i,j} + \theta, \quad (i,j)\epsilon S_2 \cup R_2 \quad ....(3.28)$$

$$\bar{c}'_{i,j} = \bar{c}_{i,j} \quad , \text{ for all other arcs } (i,j) \quad ...(3.29)$$

There will be at least one arc for which $\bar{c}'_{i,j} = 0$. The unlabeled node of that arc can now be labelled and additional labelling procedure is attempted from this node, which has two possible outcomes:

- breakthrough, which is followed by the change of flow along flow augmenting path
- non-breakthrough, followed by another potential change.

Once arc (s,t) is brought in-kilter, the algorithm continues to search for other out-of-kilter arcs and the same procedure is applied in order to bring such arcs in-kilter. If all arcs are eventually brought in-kilter, an optimal solution is found.

The potential change procedure can be utilized only a finite number of times to bring an arc (s,t) in-kilter, because each potential change reduces the number of arcs in set S for at least one. Since the number of arcs in the network is finite, there may be no arcs left in set S after a number of potential changes. At this point, the flows and node potentials cannot be changed without violation of the kilter status of at least one arc and the problem is declared infeasible.

Infeasibility implies that some of the constraints are not properly formulated. For example, if an arc has its lower bound greater than the upper, it will be impossible to fit any flow within such bounds. Another example is if arc bounds for arcs associated with a particular node are specified in such a way that the sum of upper bounds for incoming arcs is less than the sum of lower bounds for outgoing arcs. In this situation there is no flow which can satisfy both the node conservation and the prescribed bounds.

4. AN IMPROVED OKA OF BARR, GLOVER AND KLINGMAN

4.1 Numerical Effort of the Labelling and Potential Change Procedure

In order to understand the significance of Barr's et al. reformulations [52], [38] the effort involved in carrying out the labelling and potential change procedures of the original OKA of Ford and Fulkerson has to be considered first.

Consider a network with the following properties:

a) There are (on average) ten arcs associated with each node in the network.

b) The starting guess for flows and node potentials is such that on average two out of ten arcs are label eligible (e.g. from each labelled node two more nodes can be labelled in the labelling procedure).

In the first cycle, for each of the ten arcs associated with the starting node the following checks are needed:

1) Determine if the other node of the arc is already labelled.[3]

2) Check the arc direction to determine which two label eligible states need to be examined.

Check the marginal cost and flow to [illegible] the [illegible]

---

erased. However, [illegible] cycles, as only the unlabeled nodes need to be labelled. Without this check node 9 in fig 3.2 would be labelled twice.

label eligible arcs.

The first check has to be performed for every arc, while checks 2) and 3) might not be needed, depending on the outcome of the first check. Therefore, check 1) is executed ten times in the first cycle, twenty times in the second cycle, forty times in the third cycle and so forth. If there are ten arcs in the flow augmenting path (i.e. ten cycles), the first check alone has to be executed 5120 times! In addition to this, the checks in 2) and 3) are applied in each cycle, which could increase the total number of checks by several orders of magnitude.

One should bear in mind that this is only a portion of the effort of bringing one arc in-kilter. Several breakthroughs and potential changes might be required to bring one out-of-kilter arc in-kilter.

The potential change procedure also involves significant computational effort. To illustrate this, consider a large network, and suppose that before each non-breakthrough approximately half of all the nodes in the network are labelled. In order to determine arcs which lead from labelled to unlabeled nodes and to classify them into sets S_1, R_1, S_2 and R_2, almost all arcs in the network have to be examined for their connectivity, and roughly half of them might belong to one of the four sets, in which case two [illegible] number of times as part of bringing one arc in-kilter.

These observations indicate that techniques such as OKA are only practical (for large networks) with high speed electronic computers.

4.2 Description of the Improved OKA of Barr et al.

This reformulation of the original OKA is based on a number of changes related to the capacitated network and primal and dual variables, which results in simplified labelling and potential change procedures.

4.2.1 Pseudo Arcs and Pseudo Network

The idea of creating a pseudo network as a convenient representation of the actual network was originally proposed by Klein [53]. In this reformulation, each arc of the actual network (i,j) is split into two pseudo arcs. The direction of the first pseudo arc, its upper bound $K_{i,j}$ and cost $c_{i,j}$ are the same as those of the original arc (i,j), but it has no lower bound (i.e. lower bound equals zero). The other pseudo arc is directed from node j to i, its cost is equal to $-c_{i,j}$ and its upper bound is equal to $-L_{i,j}$ while its lower bound is also equal to zero. The flow on the first pseudo arcs is equal to the flow of the original arc, while the flow on the other pseudo arcs is equal to the negative of the flow on the original arc. Each time the flow or marginal [illegible] direction as the original arc as the "main", and the other as the "mirror".

As a result of this, one can chose to consider only pseudo arcs directed out of a node. This eliminates the need to check for arc direction in the labelling and potential change procedures; also, this kind of pseudo network allows for introduction of further enhancements.

4.2.2 Net Capacity and Marginal Cost

Since each pseudo arc has a lower bound of zero, define net capacity as a variable which indicates the ability of a pseudo arc to accept a flow increase without exceeding its upper bound. For each arc (i,j) of the original network net capacities of the main and the mirror pseudo arc are defined as follows:

$$nc'_{i,j} = K_{i,j} - x_{i,j}$$

$$nc''_{i,j} = x_{i,j} - L_{i,j}$$

The convenience of using net capacity as a primal variable can be summarized in the following:

1) There is no need to check the flow against the bounds in the labelling and potential change procedures, since net capacity contains this information in its own value.
2) For a feasible circulation, all net capacities will be non-negative (i.e. they satisfy the non-negativity constraints).
3) The minimum cost flow problem can be solved by finding

each time net capacity is changed on one pseudo arc, the exact opposite amount of change has to be applied on the other.

A similar rule exists for changing marginal cost; change on one pseudo arc automatically requires the same change on the other. Rather then calculating marginal cost indirectly through the value of node potentials, marginal cost is associated with each pseudo arc. The advantage of this becomes more obvious when the new potential change procedure is formulated.

4.2.3 New Labelling Procedure

In order to apply the new labelling procedure successfully, an efficient data organization scheme is required. This is briefly outlined in the following paragraphs.

1) *Pseudo arc number*. Fast and easy access from a main pseudo arc to its mirror is essential for efficiency of this algorithm. There are several ways to achieve this; the one proposed by Barr et al. saves arrays related to the original network. Assign an integer value (arc number) $a_{i,j}$ to all main pseudo arcs in the same way in which the original arcs were numbered, while all mirrors are numbered $(a_{i,j}+n)$, where n is the maximum number of arcs in the original network. If a pseudo arc's number is b, it ranges from 1 to 2n, where

be expressed as functions of b:

$$b \le n \quad \rightarrow \quad \bar{c}(b) = \bar{c}_{i,j}$$

$$b > n \quad \rightarrow \quad \bar{c}(b) = -\bar{c}_{i,j}$$

$$b \le n \quad \rightarrow \quad nc(b) = nc'_{i,j}$$

$$b > n \quad \rightarrow \quad nc(b) = nc''_{i,j}$$

$$b \le n \quad \rightarrow \quad m(b) = b + n$$

$$b > n \quad \rightarrow \quad m(b) = b - n$$

The mirror function m(b) gives the mirror arc number for any pseudo arc b. Each time flow (or marginal cost) is changed on one pseudo arc, the same changes with opposite sign have to be implemented on its mirror. Therefore, if changes are introduced on an arc which is designated as a mirror arc (b>n), the mirror function applied to this arc indicates that "mirror of the mirror" is the main arc. Thus flow or marginal cost changes occur on both pseudo arcs simultaneously, which is equivalent to changes on one arc in the original network.

Following these observations, it is possible to rewrite the in-kilter conditions (3.23)-(3.25) and the label eligible conditions (λ, μ, ν and ρ) using the new network structure and the foregoing functions. The new in-kilter conditions are:

$$\bar{c}(b) > 0 \; ; \qquad nc[m(b)] = 0 \; \ldots\ldots\ldots\ldots (3.23')$$

$$[illegible] \qquad nc(b) > 0 \quad nc[m(b)] > 0 \; \ldots\ldots (3.24')$$

Note that if $\bar{c}(b) > 0$ then $\bar{c}(m(b)) < 0$ and vice versa.

The new labelling procedure proceeds as follows: Starting from a labelled node i, consider only pseudo arcs which are directed out of node i and label their (unlabeled) head nodes if they satisfy one of the following conditions:

λ: $\bar{c}(b) > 0$ and $nc(m(b)) < 0$

μ: $\bar{c}(b) \leq 0$ and $nc(b) > 0$

ν: $\bar{c}(b) \leq 0$ and $nc(b) > 0$

ρ: $\bar{c}(b) > 0$ and $nc(m(b)) < 0$

Original conditions λ and ρ have become identical, as well as conditions μ and ν. Thus only two label eligible conditions need to be examined for each pseudo arc directed out of node i. Moreover, if all pseudo arcs directed out of node i are subdivided into subsets of label eligible and ineligible arcs (P_i and Q_i, respectively), there is no need to examine their label eligibility every time labelling procedure is applied. However, changing flow or marginal cost can change the label eligibility of some pseudo arcs; these changes have to be recorded in subsets P_i and Q_i. Barr et al. call this *partitioning technique*, and they apply it in both the labelling and potential change procedures.

The new labelling procedure consists of two checks: for a labelled node i check if its P_i set is empty, and if not, check if head nodes of pseudo arcs in set P_i are already labelled. If not, each unlabeled node j is assigned a label which now contains only pseudo arc number b. Generation of labelled node i access its set P_i and label eligible pseudo arcs leading out of that node are at hand. To summarize, simplification of the labelling procedure is a result of:

1) Network reformulation, which eliminates checking of arc direction.
2) Net capacity as a primal variable, which eliminates checking of flow against the bounds.
3) Partitioning technique, which reduces label eligibility checks to the minimum.

The new labelling procedure results in breakthrough or non-breakthrough, similar to the original labelling procedure of Ford and Fulkerson's OKA.

In *Breakthrough* section net capacities of pseudo arcs that lie in the flow augmenting path are compared to determine the pseudo arc whose net capacity is closest to zero. [3]

All arcs in the flow augmenting path belong to label eligible states λ or ν, which can be easily determined by inspection of their marginal cost. If net capacities of arcs in states ν and λ are denoted with nc(b) and nc(m(b)), respectively, then:

$$\alpha = \min\,[\mathrm{nc}(b),\ \mathrm{abs}(\mathrm{nc}(m(b)))]$$

Having identified minimum permissible flow change α, change net capacities of each pseudo arc (and its mirror) that lies in the flow augmenting path as follows:

$nc(b) = nc(b) - a$, for arcs in state ν

$nc(m(b)) = nc(m(b)) + a$, for arcs in state λ

Equivalence to the original four label eligible conditions and desired flow changes is established as follows: Consider the flow augmenting path in fig 1.2. All label eligible arcs belong to one of the four label eligible states λ, μ, ν, and ρ. Suppose that the same nodes are labelled using the pseudo network representation. Arcs 19 and 18 are represented by the main pseudo arcs, while 17 and 20 are represented by their mirrors (this is due to considering only out-going pseudo arcs in the labelling procedure). One can compare the desired action on each label eligible arc in the original network with action on its corresponding pseudo arc:

a) State μ: desired action on the original arc is increase of flow $x_{i,j}$, which is, in this case, equivalent to decrease in net capacity nc(b), since:

$$nc(b) = K_{i,j} - x_{i,j}$$

b) State ν: desired action on the original arc is decrease in flow $x_{i,j}$. In flow augmenting path of the corresponding pseudo network, this arc will be represented by its mirror pseudo arc, with net capacity defined as:

$$nc(b) = x_{i,j} - L_{i,j}$$

c) State λ: desired action is increase of flow $x_{i,j}$. Note that this arc is represented by its main pseudo arc in flow augmenting path, and net capacity of its mirror arc is defined as:

$$nc(m(b)) = x_{i,j} - L_{i,j}$$

where any increase in flow would also increase the net capacity nc(m(b)).

d) State ρ: desired action is decrease of flow $x_{i,j}$. In the flow augmenting path of the pseudo network this arc will be represented by its mirror with $nc(b)=x_{i,j}-L_{i,j}$, while the corresponding net capacity of the "mirror of the mirror" is equal to:

$$nc(m(b)) = K_{i,j} - x_{i,j}$$

Again, decrease in flow results in increase in net capacity of the mirror arc.

Thus the equivalence of flow change in Ford-Fulkerson's OKA and change of net capacities in Barr's OKA is fully established.

Changes of net capacities might affect label eligibility of some arcs and sets P and Q have to be updated accordingly. One can distinguish between pseudo arcs which change from label eligible into label ineligible, and vice versa:

1) *Termination of label eligibility.*

There are two possibilities:

a) If a pseudo arc is in state ν and its new net capacity is zero, it is no longer possible to decrease its net capacity without violating its kilter status and it should be moved from P into Q.

b) By the same reasoning, a pseudo arc in state λ whose new mirror capacity nc(m(b)) becomes zero should be moved from P into Q.

2) *Changing from ineligibility into label eligibility.* Pseudo arcs that lie in the flow augmenting path are not affected by this change, since they are all label eligible before the flow change. In the same time,their mirrors might not be label eligible, and these are the only arcs which may become label eligible after the flow change. To extend this observation, note that each pseudo arc which is in state λ will have its mirror in state ν. The opposite correspondence also exists, except for the case when marginal cost of the pseudo arc in state ν is zero. In this case, if a pseudo arc that lies in the flow augmenting path has its marginal cost equal to zero and net capacity positive (ν), its mirror will also have marginal cost equal to zero. Any decrease in net capacity on the main pseudo arc means simultaneous increase in net capacity of its mirror; if this new mirror capacity becomes positive the mirror arc will join state ν, and it should be moved from set Q into set P.

4.2.4 Potential Change Procedure

As in the case of the original Ford-Fulkerson's OKA, the new potential change procedure is aimed at extending the labelling tree, which may eventually result in a breakthrough. The first thing to do in the original potential change procedure was to determine arcs whose (only) one node was labelled. Use of the existing partition has resulted in considerable simplification of this step in the new potential change procedure. For each labelled node i, examine only pseudo arcs in set Q_i, since only such arcs can lead to unlabeled nodes. Moreover, the conditions of membership of sets S_1 and S_2 have become identical due to the symmetrical properties of the pseudo network. Note that if an arc in the original network belongs to either S_1 or S_2, its corresponding pseudo arc b will have the following properties:

$$\bar{c}(b) > 0 \text{ and } nc(b) \geq 0$$

One can observe the following steps in the new potential change procedure:

1) Access all arcs in subsets Q_i associated with each labelled node i, and classify them into sets S and R. If there are no arcs in set S, declare the problem infeasible and stop.
2) Compare marginal cost of all pseudo arcs in set S and isolate the one which is closest to zero. Marginal cost of this arc will determine the amount of marginal cost change θ as:

   $$\theta = \min \bar{c}(b), \quad b \epsilon S$$

3) Change marginal cost of each pseudo arc in set Q=SUR as follows:

   $$\bar{c}(b) = \bar{c}(b) - \theta \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.26')$$

   while in the same time mirror of each of these arcs undergoes the same change with opposite sign:

$$\bar{c}(m(b)) = \bar{c}(m(b)) + \theta \text{ ....} (3.27')$$

All other pseudo arcs will have their marginal cost unchanged.

Having completed one potential change, there is at least one more labelled node. New labelling procedure is now attempted from this node, with possibilities to perform breakthrough, followed by a flow change, or non-breakthrough, followed by another potential change, and so on. In that respect Barr's algorithm follows exactly the same logic of Ford and Fulkerson's algorithm.

Marginal cost change can also affect label eligibility and subsets P_i and Q_i have to be updated accordingly. There are two possible shifts:

1) If $b \in S$, $nc(b)>0$ and new marginal cost $\bar{c}(b)=0$, then arc b has entered state ν and it should be moved from P to Q.
2) If $b \in Q$, $nc(b)=0$, the old marginal cost was non-negative and the new marginal cost is $\bar{c}(b)<0$, then m(b) becomes label ineligible and it should be moved from P to Q.

Condition 1) is straight forward and needs no further comments. However, condition 2) is not obvious and it will be briefly discussed here. Note that $\bar{c}(b)<0$ implies that $\bar{c}(m(b))>0$. In order to have arc m(b) label eligible, its mirror (arc b) must have its net capacity positive, which is not the case, since $nc(b) = 0$. Therefore, if arc m(b) was label eligible before the potential change it has to be moved from set P into set Q.

Finally, the new potential change procedure has one more important feature which does not exist in the original OKA. As previously mentioned, more than one potential change might be needed in order to bring one arc in-kilter. If that happens, Barr's algorithm offers the possibility to save results of previous potential change and therefore minimize the effort of completing any subsequent potential changes. This is made possible as a combined result of two factors:

a) Marginal cost is associated with each pseudo arc.

b) Partition is updated with each potential change.

This means that once sets S and R are established in one potential change, one only needs to examine set(s) Q_i of the new labelled node(s) in the second potential change and extend the existing sets S and R (if set Q_i is not empty).

The reason this couldn't be done in the original OKA lies in the fact that there was no mechanism to keep track of label ineligibility of arcs, i.e. there was no partition and all arcs had to be examined for membership in sets S and R in every potential change. Barr et al. call this "recycling" of sets S and R. The effort to complete multiple potential changes is thus exponentially reduced in case of Barr's algorithm.

Barr's algorithm also ends when all pseudo arcs are brought in-kilter or when potential change procedure cannot be performed since set S is empty. In the first case, values of flow $x_{i,j}$ on original arcs can be retrieved using one of the two equations for net capacities.

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN        , DATE: MAY 02, 1988    TIME: 15:43:19      NAME: ARRAYS  PAGE:    23
                    *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN        34         IC=JNODE(L)
ISN        35         JNODE(L)=IC+1
ISN        36         ELSE
ISN        37         IC=MIDP(L)
ISN        38         MIDP(L)=IC-1
ISN        39         ENDIF
ISN        40         NDARCS(IC)=M
ISN        41    6    CONTINUE
ISN        42   11    CONTINUE
ISN        43    9    CONTINUE
ISN        44         RETURN
ISN        45         END
*STATISTICS*    SOURCE STATEMENTS = 45, PROGRAM SIZE = 3500 BYTES, PROGRAM NAME = ARRAYS.    PAGE:   22.
*STATISTICS*     NO DIAGNOSTICS GENERATED.
**ARRAYS** END OF COMPILATION 7 ******
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:43:22                    PAGE:    60
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK    SOURCE    TERM   OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
                    NOSYM NORENT   SDUMP AUTODBL(NONE)   NOSXM    IL
                      OPT(0) LANGLVL(77) NOFIPS    FLAG(I)  NAME(MAIN   )  LINECOUNT(60)       CHARLEN(500)
                *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN       1           SUBROUTINE LBLRTN(IT,IS,MARCS2,MND,MND1,NNLS1,NLN,NNLS,NEXNOD,
                     +NSETS,NSTS1,NODP,MIDP,NDARCS,NLOWST,NHIST,LABEL,INLOC,NHEAD,*)
                C...SUBROUTINE ATTEMPTS TO LABEL A FLOW AUGMENTING PATH.
ISN       2           INTEGER*4 LABEL(MND),INLOC(MND),MIDP(NSETS),NODP(NSTS1),
                     +NDARCS(MARCS2),NHEAD(MARCS2),NLOWST(MND),NHIST(MND),NEXNOD(NSETS)
ISN       3           DO 1 NNLS=NNLS1,MND
ISN       4           I=INLOC(NNLS)
ISN       5           NSTRT=NLOWST(I)
ISN       6           NEND=NHIST(I)
ISN       7           DO 2 MV=NSTRT,NEND
ISN       8           IF(LABEL(NEXNOD(MV)).NE.0)GO TO 2
ISN       9           IF(NODP(MV).LE.MIDP(MV))THEN
ISN      10           JP=NODP(MV)
ISN      11           JARC=NDARCS(JP)
ISN      12           NEWNOD=NHEAD(JARC)
ISN      13           LABEL(NEWNOD)=JARC
ISN      14           NLN=NLN+1
ISN      15           INLOC(NLN)=NEWNOD
ISN      16           IF(NEWNOD.EQ.IS)GO TO 4
ISN      17           ELSE
ISN      18           ENDIF
ISN      19     2     CONTINUE
ISN      20           IF(NNLS.EQ.NLN)GO TO 5
ISN      21     1     CONTINUE
                C...REGULAR RETURN INDICATES  BREAKTHROUGH.
ISN      22     4     RETURN
                C...LABELLED RETURN INDICATES  POTENTIAL CHANGE.
ISN      23     5     RETURN 1
ISN      24           END
*STATISTICS*    SOURCE STATEMENTS = 24, PROGRAM SIZE = 2192 BYTES, PROGRAM NAME = LBLRTN.    PAGE:   60.
*STATISTICS*     NO DIAGNOSTICS GENERATED.
**LBLRTN** END OF COMPILATION 27 ******
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:43:26                 PAGE:    95
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE   TERM   OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
                    NOSYM NORENT   SDUMP AUTODBL(NONE)   NOSXM   IL
                      OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)        CHARLEN(500)
                 *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN        1           SUBROUTINE PINQ(NSN1,NPA,MARCS2,MARCS,NSETS,NSTS1,NSETA,NEXNOD,
                      +SETMAP,CHMAP,MIDP,NODP,NDARCS)
                 C...SUBROUTINE MOVES AN ARC FROM SUBSET P INTO SUBSET Q.
ISN        2           INTEGER*4 MIDP(NSETS),NODP(NSTS1),NDARCS(MARCS2),NSETA(MARCS),
                      +NEXNOD(NSETS),SETMAP(NSETS),CHMAP(321,5)
ISN        3           IF(NPA.GT.MARCS)THEN
ISN        4           NAC=NPA-MARCS
ISN        5           ELSE
ISN        6           NAC=NPA
ISN        7           ENDIF
ISN        8           NSC=NSETA(NAC)
ISN        9           IF(NEXNOD(NSC).EQ.NSN1)NSC=CHMAP(SETMAP(NSC),5)
ISN       11           MSTART=NODP(NSC)
ISN       12           MEND=MIDP(NSC)
ISN       13           IF(MSTART.GT.MEND)RETURN
ISN       15           DO 1 IAC=MSTART,MEND
ISN       16           IF(NDARCS(IAC).EQ.NPA)THEN
ISN       17           ND1=NDARCS(MEND)
ISN       18           NDARCS(MEND)=NPA
ISN       19           NDARCS(IAC)=ND1
ISN       20           MIDP(NSC)=MEND-1
ISN       21           ELSE
ISN       22           ENDIF
ISN       23     1     CONTINUE
ISN       24           RETURN
ISN       25           END
*STATISTICS*    SOURCE STATEMENTS = 23, PROGRAM SIZE = 1896 BYTES, PROGRAM NAME = PINQ   .     PAGE:    95.
*STATISTICS*     NO DIAGNOSTICS GENERATED.
**PINQ** END OF COMPILATION 41 ******
```

```
LEVEL 1.4.1 (MAY 1985)           VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:43:27                    PAGE:  109
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE   TERM   OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
                    NOSYM NORENT   SDUMP AUTODBL(NONE)   NOSXM   IL
                      OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)        CHARLEN(500)
                    *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN       1               SUBROUTINE PSARCS(ACOST,MARCS,MND,
                         +MARCS2,MND1,Z,NSETS,NSTS1,NTAIL,NHEAD,JNODE,PACOST,
                         +NODARC,NCHA,NSETA,SETMAP,MIDP,NODP,NDARCS,NLOWST,NHIST)
                    C...SUBROUTINE TRANSFORMS THE ORIGINAL CAPACITATED NETWORK INTO
                    C..." PSEUDO-ARCS " NETWORK AS REQUIRED BY THE REFORMULATED OUT-
                    C...OF-KILTER ROUTINE.
ISN       2               INTEGER*4 NTAIL(MARCS2),NHEAD(MARCS2),NODP(NSETS),SETMAP(NSETS),
                         +MIDP(NSETS),JNODE(MARCS2),NDARCS(MARCS2),PACOST(MARCS2),
                         +NCHA(321),NODARC(NSETS),NLOWST(MND),NHIST(MND),NSETA(MARCS)
ISN       3               REAL*8 Z,ACOST(MARCS)
                    C...THIS SECTION CREATES PSEUDO ARCS.
ISN       4               DO 3 M=1,MARCS
ISN       5               N=M+MARCS
ISN       6               PACOST(M)=IDNINT(ACOST(M)*Z)
ISN       7               PACOST(N)=-PACOST(M)
ISN       8               I=NTAIL(M)
ISN       9               J=NHEAD(M)
ISN      10               NHEAD(N)=I
ISN      11               NTAIL(N)=J
ISN      12     3         CONTINUE
ISN      13               NODP(NSTS1)=MARCS2+1
ISN      14               NSUM=1
ISN      15               KK=1
ISN      16               DO 4 I=1,MND
ISN      17               NSTRT=NLOWST(I)
ISN      18               NEND=NHIST(I)
ISN      19               DO 5 L=NSTRT,NEND
ISN      20               LMIN=NODARC(L)
ISN      21               LMAX=LMIN+NCHA(SETMAP(L))-1
ISN      22               NODP(L)=NSUM
ISN      23               JNODE(L)=NSUM
ISN      24               NSUM=NCHA(SETMAP(L))+NSUM
ISN      25               MIDP(L)=NSUM-1
ISN      26               DO 10 M=LMIN,LMAX
ISN      27               IF(M.LT.KK)GO TO 10
ISN      28               NSETA(M)=L
ISN      29               KK=KK+1
ISN      30    10         CONTINUE
ISN      31     5         CONTINUE
ISN      32     4         CONTINUE
ISN      33               RETURN
ISN      34               END
*STATISTICS*    SOURCE STATEMENTS = 34, PROGRAM SIZE = 2908 BYTES, PROGRAM NAME = PSARCS.    PAGE:  109.
*STATISTICS*     NO DIAGNOSTICS GENERATED.
**PSARCS** END OF COMPILATION 44 ******
```

```
LEVEL 1.4.1 (MAY 1985)          VS FORTRAN          DATE: MAY 02, 1988    TIME: 15:43:27                PAGE:  110
OPTIONS IN EFFECT:  NOLIST NOMAP NOXREF NOGOSTMT NODECK   SOURCE   TERM   OBJECT FIXED  NOTEST NOTRMFLG   SRCFLG
                    NOSYM NORENT   SDUMP AUTODBL(NONE)   NOSXM   IL
                      OPT(0) LANGLVL(77) NOFIPS   FLAG(I)  NAME(MAIN  )  LINECOUNT(60)       CHARLEN(500)
                 *....*...1.........2.........3.........4.........5.........6.........7.*.......8
ISN        1           SUBROUTINE QINP(NSN,NMA,MARCS2,MARCS,NSETS,NSTS1,NSETA,NEXNOD,
                      +SETMAP,CHMAP,MIDP,NODP,NDARCS)
                 C...SUBROUTINE MOVES AN ARC FROM SUBSET Q INTO SUBSET P.
ISN        2           INTEGER*4 MIDP(NSETS),NODP(NSTS1),NDARCS(MARCS2),NSETA(MARCS),
                      +NEXNOD(NSETS),SETMAP(NSETS),CHMAP(321,5)
ISN        3           IF(NMA.GT.MARCS)THEN
ISN        4           NAC=NMA-MARCS
ISN        5           ELSE
ISN        6           NAC=NMA
ISN        7           ENDIF
ISN        8           NSC=NSETA(NAC)
ISN        9           IF(NEXNOD(NSC).EQ.NSN)NSC=CHMAP(SETMAP(NSC),5)
ISN       11           MSTART=MIDP(NSC)+1
ISN       12           MEND=NODP(NSC+1)-1
ISN       13           DO 1 IAC=MSTART,MEND
ISN       14           IF(NDARCS(IAC).EQ.NMA)THEN
ISN       15           ND1=NDARCS(MSTART)
ISN       16           NDARCS(MSTART)=NMA
ISN       17           NDARCS(IAC)=ND1
ISN       18           MIDP(NSC)=MSTART
ISN       19           ELSE
ISN       20           ENDIF
ISN       21     1     CONTINUE
ISN       22           RETURN
ISN       23           END
*STATISTICS*    SOURCE STATEMENTS = 22, PROGRAM SIZE = 1868 BYTES, PROGRAM NAME = QINP   .     PAGE:   110.
*STATISTICS*     NO DIAGNOSTICS GENERATED.
**QINP** END OF COMPILATION 45 ******
```

APPENDIX III

This appendix contains SCF file for two experimental runs of WRMM, in which only reservoir penalties were changed. The SCF file listing is followed by the output from subroutine arcchk for the first two time intervals, and then the final output is presented. The final output demonstrates how change in penalties results in different solution by the model. Maintaining constant reservoir storage was given the highest priority in the second run and it remains constant throughout the simulation.

```
REFNA   ILICH
STUDY
SAMPLE PROBLEM FOR OKA INVESTIGATION
SYSTE
FICTICTIOUS
RUN
SAMPLE PRIORITY SYSTEM
USER    NEBOJSA ILIC
DATE    JUNE 87
REMAR   A 1 YEAR RUN (1987), USING MONTHLY INTERVALS
TITLE   SAMPLE PROBLEM FOR OKA INVESTIGATION


$SIMC
INTRV        12    0    0
  31.  28.  31.  30.  31.  30.  31.  31.  30.  31.  30.  31.
CYCLE         1    0    0
APPTP         1    0    0
   12
ADJIN         4    0    0
    3    3    3    3
START         1 1987    0
DEBUG         0    0    0
ABE                                                              1    3
OUTEV         0    0    0
$PHYS         0    0    0
RESER        61  LAKE                        3    1    0            0.0      0.00
       0.0    1000.0  100000.0    1020.0  300000.0    1030.0
  101 1.000
IRRIG        71  TRACT                       1    0    0        10000.0      1.00
   21   52 1.00
WITHD        81   TOWN                       1    0    0            0.5      0.00
   22   52 1.00
NATCH         1  RIV1                       61   52    0            0.0      0.00
NATCH         2  STREAM                     51   52    0            0.0      0.00
APPTC        41  RIVAPP                     52    0    0            0.0      0.00
      0.30     13.00      0.00    0
DIVCH        11  DIVERS                     61   71    0         1000.0      0.00
DIVCH        12  DIVSTR                     51   81    0         1000.0      0.00
JUNCT        51  UP1                         0    0    0            0.0      0.00
JUNCT        52  FORK                        0    0    0            0.0      0.00
CTLST         0  CONTROL                     1   61    3   CC       0.0      0.00
       0.0    1000.0      30.0    1020.0     100.0    1030.0
$PENS         0                              0    0    0            0.0      0.00
RESER    1                        1     1    3             0.        0.
      5.00
     10.00     20.00    100.00
   61              3
    1    1030.5    1030.0    1027.0    1025.0
  121    1030.5    1030.0    1028.0    1025.0
  244    1030.5    1030.0    1028.0    1025.0
IRRIG    1                        1     3    0             0.        0.
     15.00     25.00     80.00
   71              1
    1       0.9       0.7       0.0
```

```
WITHD     1                        1    2    0              0.        0.
   81
      0.80      0.00
     50.00     60.00
NATCH     1                        2    0    1              0.        0.
    18.00
    1         1
    1    1000.0    20.0
    2         1
    1    1000.0    20.0
APPTC     1                        1    1    0              0.        0.
   41
      0.00      0.00     10.00     55.00      0.00      0.00    100.00
DIVCH     1                        2    0    0              0.        0.
    200.00     0.00
   11         3  200000000.0
    1    1000.0      0.0        0.0
   91    1000.0    1000.0       0.0
  305    1000.0      0.0        0.0
   12         1  200000000.0
    1    1000.0    1000.0       0.0
$WATD    0                         0    0    0              0.        0.
IRRIG         71                C C C
    0.0000    0.0000    0.0000   30.0000   60.0000  130.0000  166.0000  104.0000
   90.0000   40.0000    0.0000    0.0000
    0.3000
    0.0000
    0.6000
MAJOR         81
    2.6000    2.7000    2.9000    4.2700    5.1200    6.1200    6.1200    2.1200
    6.1200    6.1200    4.0000    4.0000
MINOR         52
    0.5000    0.5000    0.5000    0.5000    0.9000    0.9000    0.9000    0.3000
    0.9000    0.5000    0.5000    0.5000
$WATS          0
INSTO         61                   0    0             1030.0000
INFLN         61                   0    0                0.0000
    5.0000    6.0000    7.0000    8.0000    9.0000   10.0000    9.0000   19.0000
    7.0000    7.0000    7.0000    5.0000
INFLN         51                   0    0                0.0000
    3.0000    4.0000    4.0000    4.0000    5.0000    7.0000    4.0000   10.0000
    3.0000    3.0000    3.0000    3.0000
APPTF         41                   0   12                0.0000
   30.0000   32.0000   40.0000   60.0000  100.0000  110.0000   90.0000   63.0000
   28.0000   31.0000   52.0000   55.0000
EVAP         101                   0    0                0.0000
    0.0000    0.0000    5.0000   17.0000   53.0000   98.0000  132.0000   75.0000
   31.0000   14.0000    0.0000    0.0000
```

```
                        --->   SUBROUTINE  ARCCHK   <---
################################################################################


                       YEAR = 1987           INTERVAL =  1

#
NATCHL     1 : ARCNO          1         2
               ARCUB    100.000    20.000
               AFLOW     20.000    11.420
               ARCLB     20.000     0.000
               ACOST      0.000    18.000
#
DIVCHL    11 : ARCNO          3         4         5
               ARCUB   1000.000     0.000     0.000
               AFLOW      0.000     0.000     0.000
               ARCLB      0.000     0.000     0.000
               ACOST    200.000     0.000     0.000
#
RESERV    61 : ARCNO          6         7         8         9        10        11
               ARCUB      3.734   112.007   112.007    20.535    16.801    74.671
               AFLOW      0.000   112.007   112.007     3.580     0.000     0.000
               ARCLB      0.000   112.007   112.007     0.000     0.000     0.000
               ACOST      5.091     0.000     0.000    10.091    20.091   100.091
#
INFLOW    61 : ARCNO         12        13
               ARCUB      5.000     0.000
               AFLOW      5.000     0.000
               ARCLB      5.000     0.000
               ACOST      0.000     0.000
#
RETURN    21 : ARCNO         14        15        16        17
               ARCUB      0.000     0.000     0.000     0.000
               AFLOW      0.000     0.000     0.000     0.000
               ARCLB      0.000     0.000     0.000     0.000
               ACOST      0.000    15.000    25.000    80.000
#
IRRIGAT   71 : ARCNO         18        19        20        21
               ARCUB      0.000     0.000     0.000     0.000
               AFLOW      0.000     0.000     0.000     0.000
               ARCLB      0.000     0.000     0.000     0.000
               ACOST      0.000    15.091    25.091    80.091
#
RETURN    22 : ARCNO         22        23        24
               ARCUB      2.600     0.520     2.080
               AFLOW      2.600     0.520     1.160
               ARCLB      2.600     0.000     0.000
               ACOST      0.000    50.000    60.000
#
NATCHL     2 : ARCNO         25        26
               ARCUB   1000.000    20.000
               AFLOW     20.000    20.000
               ARCLB     20.000     0.000
               ACOST      0.000    18.000
```

```
#
APPORT    41 : ARCNO         27         28         29         30
               ARCUB  10000.000      9.000      0.000      9.000
               AFLOW      0.000      9.000      0.000      0.000
               ARCLB      0.000      9.000      0.000      0.000
               ACOST     10.000      0.000     55.000    100.000
#
MINOR     52 : ARCNO         31
               ARCUB      0.500
               AFLOW      0.500
               ARCLB      0.500
               ACOST      0.000
#
DIVCHL    12 : ARCNO         32         33         34
               ARCUB      0.000   1000.000   1000.000
               AFLOW      0.000   1000.000    997.000
               ARCLB      0.000   1000.000      0.000
               ACOST    200.000      0.000      0.000
#
MAJOR     81 : ARCNO         35         36         37
               ARCUB      2.600      0.520      2.080
               AFLOW      2.600      0.520      0.000
               ARCLB      2.600      0.000      0.000
               ACOST      0.000     50.091     60.091
#
INFLOW    51 : ARCNO         38         39
               ARCUB      3.000      0.000
               AFLOW      3.000      0.000
               ARCLB      3.000      0.000
               ACOST      0.000      0.000
#
SYSTEM  1000 : ARCNO         40         41
               ARCUB  10000.000      0.000
               AFLOW      8.000      0.000
               ARCLB      0.000      0.000
               ACOST      0.000      0.000

################################################################################
```

```
                 --->   SUBROUTINE  ARCCHK   <---
##############################################################################

                 YEAR = 1987          INTERVAL =  2

#
NATCHL     1 : ARCNO          1          2
               ARCUB     96.644     20.000
               AFLOW     20.000     11.740
               ARCLB     20.000      0.000
               ACOST      0.000     18.000
#
DIVCHL    11 : ARCNO          3          4          5
               ARCUB   1000.000      0.000      0.000
               AFLOW      0.000      0.000      0.000
               ARCLB      0.000      0.000      0.000
               ACOST    200.000      0.000      0.000
#
RESERV    61 : ARCNO          6          7          8          9         10         11
               ARCUB      4.134    124.008    120.044     20.806     20.530     82.672
               AFLOW      0.000    124.008    120.044      6.224      0.000      0.000
               ARCLB      0.000    124.008    120.044      0.000      0.000      0.000
               ACOST      5.091      0.000      0.000     10.091     20.091    100.091
#
INFLOW    61 : ARCNO         12         13
               ARCUB      6.000      0.000
               AFLOW      6.000      0.000
               ARCLB      6.000      0.000
               ACOST      0.000      0.000
#
RETURN    21 : ARCNO         14         15         16         17
               ARCUB      0.000      0.000      0.000      0.000
               AFLOW      0.000      0.000      0.000      0.000
               ARCLB      0.000      0.000      0.000      0.000
               ACOST      0.000     15.000     25.000     80.000
#
IRRIGAT   71 : ARCNO         18         19         20         21
               ARCUB      0.000      0.000      0.000      0.000
               AFLOW      0.000      0.000      0.000      0.000
               ARCLB      0.000      0.000      0.000      0.000
               ACOST      0.000     15.091     25.091     80.091
#
RETURN    22 : ARCNO         22         23         24
               ARCUB      2.700      0.540      2.160
               AFLOW      2.700      0.540      0.320
               ARCLB      2.700      0.000      0.000
               ACOST      0.000     50.000     60.000
#
NATCHL     2 : ARCNO         25         26
               ARCUB   1000.000     20.000
               AFLOW     20.000     20.000
               ARCLB     20.000      0.000
               ACOST      0.000     18.000
```

```
#
APPORT    41 : ARCNO         27         28         29         30
               ARCUB  10000.000      9.600      0.000      9.600
               AFLOW      0.000      9.600      0.000      0.000
               ARCLB      0.000      9.600      0.000      0.000
               ACOST     10.000      0.000     55.000    100.000
#
MINOR     52 : ARCNO         31
               ARCUB      0.500
               AFLOW      0.500
               ARCLB      0.500
               ACOST      0.000
#
DIVCHL    12 : ARCNO         32         33         34
               ARCUB      0.000   1000.000   1000.000
               AFLOW      0.000   1000.000    996.000
               ARCLB      0.000   1000.000      0.000
               ACOST    200.000      0.000      0.000
#
MAJOR     81 : ARCNO         35         36         37
               ARCUB      2.700      0.540      2.160
               AFLOW      2.700      0.540      0.000
               ARCLB      2.700      0.000      0.000
               ACOST      0.000     50.091     60.091
#
INFLOW    51 : ARCNO         38         39
               ARCUB      4.000      0.000
               AFLOW      4.000      0.000
               ARCLB      4.000      0.000
               ACOST      0.000      0.000
#
SYSTEM  1000 : ARCNO         40         41
               ARCUB  10000.000      0.000
               AFLOW     10.000      0.000
               ARCLB      0.000      0.000
               ACOST      0.000      0.000

################################################################################
```

```
                                SAMPLE PROBLEM FOR OKA INVESTIGATION                                               JUNE 87
                                                                                                                   ILIC
                                                                                                                   NEBOJSA ILIC
  * * UNITS : RESERVOIR (M), IRRIGATION (MM), HYDRO (MW), ALL OTHERS (CMS)                                              PAGE  1
    COMPONENT      REF                                       TIME INTERVAL ENDING                                 9/ 4/88  18.58
  TYPE       NAME   NO.    31JAN87   28FEB87   31MAR87   30APR87   31MAY87   30JUN87   31JUL87   31AUG87   30SEP87   31OCT87   30NOV87   31DEC87
*****************************************************************************************************************************************************

RESERV   LAKE       61 1029.52A 1029.25A 1028.73A 1027.57B 1025.00B'1025.00B'1024.75C 1025.00B 1025.00B'1025.00B'1024.55C 1023.81C
IRRIGAT  TRACT      71    0.00      0.00      0.00     27.00A'   42.00B'   36.45C     0.00C'   72.80B'    7.17C     1.53C     0.00      0.00
RETURN   .........  21    0.00      0.00      0.00      0.74A'    1.60      3.22A'    3.98A'    2.50A'    2.23A'    0.96A'    0.00      0.00
MAJOR     TOWN      81    2.08A'    2.16A'    2.32A'    0.58B     0.00B'    0.00B'    0.00B'    0.00B'    0.00B'    0.00B'    0.00B'    0.00B'
RETURN   .........  22    0.92B     1.84B     1.68B     3.42A'    5.00A     6.12      4.00B     2.12      3.00B     3.00B     3.00B     3.00B
NATCHL   RIV1        1    8.58A     8.26A    10.82A    14.34A    23.59      3.68A     5.92A     9.53A     4.07A     5.84A    10.50A    10.50A
NATCHL   STREAM      2    0.00A'    0.00A'    0.00A'    0.00A'    0.00A'    0.88A     0.00A'    7.88A     0.00A'    0.00A'    0.00A'    0.00A'
APPORT   RIVAPP     41    9.00      9.60     12.00     18.00     29.29A    13.00A    13.00A    21.72A     8.40A     9.30A    13.00A    13.00A
DIVCHL   DIVERS     11    0.00      0.00      0.00      2.48A     4.21A     5.57A     3.98A     7.03A     2.69A     1.06A     0.00      0.00
DIVCHL   DIVSTR     12    3.00A     4.00A     4.00A     4.00A     5.00A     6.12A     4.00A     2.12A     3.00A     3.00A     3.00A     3.00A
EVAP-RES LAKE       61    0.000     0.000     0.037     0.131     0.396     0.756     0.986     0.560     0.239     0.105     0.000     0.000
```

```
REFNA   ILICH
STUDY
SAMPLE PROBLEM FOR OKA INVESTIGATION
SYSTE
FICTICTIOUS
RUN
SAMPLE PRIORITY SYSTEM
USER    NEBOJSA ILIC
DATE    JUNE 87
REMAR   A 1 YEAR RUN (1987), USING MONTHLY INTERVALS
TITLE   SAMPLE PROBLEM FOR OKA INVESTIGATION


$SIMC
INTRV        12    0    0
  31.  28.  31.  30.  31.  30.  31.  31.  30.  31.  30.  31.
CYCLE         1    0    0
APPTP         1    0    0
   12
ADJIN         4    0    0
    3    3    3    3
START         1 1987    0
DEBUG         0    0    0
ABE                                                            1    3
OUTEV         0    0    0
$PHYS         0    0    0
RESER        61  LAKE                      3    1    0            0.0      0.00
     0.0    1000.0  100000.0      1020.0  300000.0      1030.0
  101 1.000
IRRIG        71  TRACT                     1    0    0        10000.0      1.00
   21   52 1.00
WITHD        81   TOWN                     1    0    0            0.5      0.00
   22   52 1.00
NATCH         1  RIV1                     61   52    0            0.0      0.00
NATCH         2  STREAM                   51   52    0            0.0      0.00
APPTC        41  RIVAPP                   52    0    0            0.0      0.00
     0.30     13.00      0.00    0
DIVCH        11  DIVERS                   61   71    0         1000.0      0.00
DIVCH        12  DIVSTR                   51   81    0         1000.0      0.00
JUNCT        51  UP1                       0    0    0            0.0      0.00
JUNCT        52  FORK                      0    0    0            0.0      0.00
CTLST         0  CONTROL                   1   61    3   CC       0.0      0.00
     0.0    1000.0      30.0    1020.0     100.0    1030.0
$PENS         0                            0    0    0            0.0      0.00
RESER   1                        1    1    3             0.        0.
     5.00
   500.00    800.00   1000.00
   61         3
    1   1030.5    1030.0    1027.0    1025.0
  121   1030.5    1030.0    1028.0    1025.0
  244   1030.5    1030.0    1028.0    1025.0
IRRIG   1                        1    3    0             0.        0.
   15.00     25.00     80.00
   71         1
    1       0.9       0.7       0.0
```

```
WITHD    1                        1    2    0              0.        0.
   81
      0.80      0.00
     50.00     60.00
NATCH    1                        2    0    1              0.        0.
      0.00
    1         1
    1    1000.0     20.0
    2         1
    1    1000.0     20.0
APPTC    1                        1    1    0              0.        0.
   41
      0.00      0.00     10.00     55.00      0.00      0.00    100.00
DIVCH    1                        2    0    0              0.        0.
    200.00      0.00
   11         3  200000000.0
    1    1000.0       0.0       0.0
   91    1000.0    1000.0       0.0
  305    1000.0       0.0       0.0
   12         1  200000000.0
    1    1000.0    1000.0       0.0
$WATD    0                        0    0    0              0.        0.
IRRIG       71                 C C C
    0.0000    0.0000    0.0000   30.0000   60.0000  130.0000  166.0000  104.0000
   90.0000   40.0000    0.0000    0.0000
    0.3000
    0.0000
    0.6000
MAJOR       81
    2.6000    2.7000    2.9000    4.2700    5.1200    6.1200    6.1200    2.1200
    6.1200    6.1200    4.0000    4.0000
MINOR       52
    0.5000    0.5000    0.5000    0.5000    0.9000    0.9000    0.9000    0.3000
    0.9000    0.5000    0.5000    0.5000
$WATS        0
INSTO       61                    0    0          1030.0000
INFLN       61                    0    0             0.0000
    5.0000    6.0000    7.0000    8.0000    9.0000   10.0000    9.0000   19.0000
    7.0000    7.0000    7.0000    5.0000
INFLN       51                    0    0             0.0000
    3.0000    4.0000    4.0000    4.0000    5.0000    7.0000    4.0000   10.0000
    3.0000    3.0000    3.0000    3.0000
APPTF       41                    0   12             0.0000
   30.0000   32.0000   40.0000   60.0000  100.0000  110.0000   90.0000   63.0000
   28.0000   31.0000   52.0000   55.0000
EVAP       101                    0    0             0.0000
    0.0000    0.0000    5.0000   17.0000   53.0000   98.0000  132.0000   75.0000
   31.0000   14.0000    0.0000    0.0000
```

```
                  --->   SUBROUTINE  ARCCHK   <---
################################################################################

                  YEAR = 1987          INTERVAL =  1

#
NATCHL     1 : ARCNO          1          2
               ARCUB    100.000     20.000
               AFLOW     25.000     20.000
               ARCLB     20.000      0.000
               ACOST      0.000      0.000
#
DIVCHL    11 : ARCNO          3          4          5
               ARCUB   1000.000      0.000      0.000
               AFLOW      0.000      0.000      0.000
               ARCLB      0.000      0.000      0.000
               ACOST    200.000      0.000      0.000
#
RESERV    61 : ARCNO          6          7          8          9         10         11
               ARCUB      3.734    112.007    112.007     20.535     16.801     74.671
               AFLOW      0.000    112.007    112.007      0.000      0.000      0.000
               ARCLB      0.000    112.007    112.007      0.000      0.000      0.000
               ACOST      5.091      0.000      0.000    500.091    800.091   1000.091
#
INFLOW    61 : ARCNO         12         13
               ARCUB      5.000      0.000
               AFLOW      5.000      0.000
               ARCLB      5.000      0.000
               ACOST      0.000      0.000
#
RETURN    21 : ARCNO         14         15         16         17
               ARCUB      0.000      0.000      0.000      0.000
               AFLOW      0.000      0.000      0.000      0.000
               ARCLB      0.000      0.000      0.000      0.000
               ACOST      0.000     15.000     25.000     80.000
#
IRRIGAT   71 : ARCNO         18         19         20         21
               ARCUB      0.000      0.000      0.000      0.000
               AFLOW      0.000      0.000      0.000      0.000
               ARCLB      0.000      0.000      0.000      0.000
               ACOST      0.000     15.091     25.091     80.091
#
RETURN    22 : ARCNO         22         23         24
               ARCUB      2.600      0.520      2.080
               AFLOW      2.600      0.000      0.000
               ARCLB      2.600      0.000      0.000
               ACOST      0.000     50.000     60.000
#
NATCHL     2 : ARCNO         25         26
               ARCUB   1000.000     20.000
               AFLOW     20.400     20.000
               ARCLB     20.000      0.000
               ACOST      0.000      0.000
```

```
#
APPORT     41 : ARCNO          27          28          29          30
                ARCUB 10000.000       9.000       0.000       9.000
                AFLOW       0.000       9.000       0.000       1.500
                ARCLB       0.000       9.000       0.000       0.000
                ACOST      10.000       0.000      55.000     100.000
#
MINOR      52 : ARCNO          31
                ARCUB       0.500
                AFLOW       0.500
                ARCLB       0.500
                ACOST       0.000
#
DIVCHL     12 : ARCNO          32          33          34
                ARCUB       0.000    1000.000    1000.000
                AFLOW       0.000    1000.000     997.400
                ARCLB       0.000    1000.000       0.000
                ACOST     200.000       0.000       0.000
#
MAJOR      81 : ARCNO          35          36          37
                ARCUB       2.600       0.520       2.080
                AFLOW       2.600       0.520       2.080
                ARCLB       2.600       0.000       0.000
                ACOST       0.000      50.091      60.091
#
INFLOW     51 : ARCNO          38          39
                ARCUB       3.000       0.000
                AFLOW       3.000       0.000
                ARCLB       3.000       0.000
                ACOST       0.000       0.000
#
SYSTEM   1000 : ARCNO          40          41
                ARCUB 10000.000       0.000
                AFLOW       8.000       0.000
                ARCLB       0.000       0.000
                ACOST       0.000       0.000

##################################################################################
```

```
                        --->   SUBROUTINE  ARCCHK   <---
#################################################################################

                       YEAR = 1987          INTERVAL =  2

#
NATCHL       1 : ARCNO         1         2
                 ARCUB   100.000    20.000
                 AFLOW    26.000    20.000
                 ARCLB    20.000     0.000
                 ACOST     0.000     0.000
#
DIVCHL      11 : ARCNO         3         4         5
                 ARCUB  1000.000     0.000     0.000
                 AFLOW     0.000     0.000     0.000
                 ARCLB     0.000     0.000     0.000
                 ACOST   200.000     0.000     0.000
#
RESERV      61 : ARCNO         6         7         8         9        10        11
                 ARCUB     4.134   124.008   124.008    20.806    20.530    82.672
                 AFLOW     0.000   124.008   124.008     0.000     0.000     0.000
                 ARCLB     0.000   124.008   124.008     0.000     0.000     0.000
                 ACOST     5.091     0.000     0.000   500.091   800.091  1000.091
#
INFLOW      61 : ARCNO        12        13
                 ARCUB     6.000     0.000
                 AFLOW     6.000     0.000
                 ARCLB     6.000     0.000
                 ACOST     0.000     0.000
#
RETURN      21 : ARCNO        14        15        16        17
                 ARCUB     0.000     0.000     0.000     0.000
                 AFLOW     0.000     0.000     0.000     0.000
                 ARCLB     0.000     0.000     0.000     0.000
                 ACOST     0.000    15.000    25.000    80.000
#
IRRIGAT     71 : ARCNO        18        19        20        21
                 ARCUB     0.000     0.000     0.000     0.000
                 AFLOW     0.000     0.000     0.000     0.000
                 ARCLB     0.000     0.000     0.000     0.000
                 ACOST     0.000    15.091    25.091    80.091
#
RETURN      22 : ARCNO        22        23        24
                 ARCUB     2.700     0.540     2.160
                 AFLOW     2.700     0.000     0.000
                 ARCLB     2.700     0.000     0.000
                 ACOST     0.000    50.000    60.000
#
NATCHL       2 : ARCNO        25        26
                 ARCUB  1000.000    20.000
                 AFLOW    21.300    20.000
                 ARCLB    20.000     0.000
                 ACOST     0.000     0.000
```

```
#
APPORT     41 : ARCNO         27         28         29         30
                ARCUB  10000.000     11.261      1.661      9.600
                AFLOW      0.000     11.261      1.661      0.100
                ARCLB      0.000     11.261      0.000      0.000
                ACOST     10.000      0.000     55.000    100.000
#
MINOR      52 : ARCNO         31
                ARCUB      0.500
                AFLOW      0.500
                ARCLB      0.500
                ACOST      0.000
#
DIVCHL     12 : ARCNO         32         33         34
                ARCUB      0.000   1000.000   1000.000
                AFLOW      0.000   1000.000    997.300
                ARCLB      0.000   1000.000      0.000
                ACOST    200.000      0.000      0.000
#
MAJOR      81 : ARCNO         35         36         37
                ARCUB      2.700      0.540      2.160
                AFLOW      2.700      0.540      2.160
                ARCLB      2.700      0.000      0.000
                ACOST      0.000     50.091     60.091
#
INFLOW     51 : ARCNO         38         39
                ARCUB      4.000      0.000
                AFLOW      4.000      0.000
                ARCLB      4.000      0.000
                ACOST      0.000      0.000
#
SYSTEM   1000 : ARCNO         40         41
                ARCUB  10000.000      0.000
                AFLOW     10.000      0.000
                ARCLB      0.000      0.000
                ACOST      0.000      0.000

################################################################################
```

```
                          SAMPLE PROBLEM FOR OKA INVESTIGATION                                                          JUNE 87
                                                                                                                        ILIC
                                                                                                                        NEBOJSA ILIC
 * * UNITS : RESERVOIR (M), IRRIGATION (MM), HYDRO (MW), ALL OTHERS (CMS)                                                      PAGE  1
    COMPONENT       REF                                      TIME INTERVAL ENDING                                      9/ 4/88  15:54
  TYPE      NAME    NO.   31JAN87  28FEB87  31MAR87  30APR87  31MAY87  30JUN87  31JUL87  31AUG87  30SEP87  31OCT87  30NOV87  31DEC87
**************************************************************************************************************************************

RESERV   LAKE       61 1030.00  1030.00  1030.00  1030.00  1030.00  1030.00  1030.00  1030.00  1030.00  1030.00  1030.00  1030.00
IRRIGAT  TRACT      71    0.00     0.00     0.00    0.00C'   0.00C'  36.45C    0.00C'  72.80B'   7.17C    1.53C    0.00     0.00
RETURN   .........  21    0.00     0.00     0.00    0.83     1.60     3.58     4.43     2.77     2.48     1.07     0.00     0.00
MAJOR     TOWN      81    0.00B'   0.00B'   0.00B'  0.00B'   0.00B'   0.00B'   0.00B'   1.70A'   0.00B'   0.00B'   0.00B'   0.00B'
RETURN   .........  22    2.60     2.70     2.90    4.00A    5.00A    6.12     4.00B    2.12     3.00B    3.00B    3.00B    3.00B
NATCHL   RIV1        1    5.00A'   6.00A'   6.96A'  7.04A'   7.00A'   3.32A'   3.59A'  11.14A'   3.82A'   5.73A'   7.00A'   5.00A
NATCHL   STREAM      2    0.40A'   1.30A'   1.10A'  0.00A'   0.00A'   0.88A    0.00A'   6.18A    0.00A'   0.00A'   0.00A'   0.00A'
APPORT   RIVAPP     41    7.50B    9.50B   10.46B  11.37B   12.70B   13.00A   11.11B   21.91A    8.40A    9.30A    9.50B    7.50B
DIVCHL   DIVERS     11    0.00     0.00     0.00    0.83A    1.60A    5.93A    4.43A    7.30A    2.94A    1.16A    0.00     0.00
DIVCHL   DIVSTR     12    2.60A    2.70A    2.90A   4.00A    5.00A    6.12A    4.00A    3.82A    3.00A    3.00A    3.00A    3.00A
EVAP-RES LAKE       61    0.000    0.000    0.037   0.131    0.396    0.756    0.986    0.560    0.239    0.105    0.000    0.000
```